Szyfry blokowe i kryptoanaliza

1. Wstęp

Niniejszy teksy stanowi podstawowe wprowadzenie do projektowania i analizy szyfru blokowego (który ma być zrozumiały dla wszystkich posiadających pewną wiedzę na temat matematyki dyskretnej). Zbadamy koncepcję szyfrów blokowych, sposobu uzyskiwania ich bezpieczeństwa i zasad związanych z ich konstrukcją. Popularna klasa szyfrów blokowych, znana jako szyfry Feistela, zostanie opisana szczegółowo. Tekst ten opisuje i demonstruje niektóre nowoczesne metody kryptoanalizy szyfrów blokowych, pokazując, w jaki sposób można je zastosować do wariantów specjalnie zaprojektowanego szyfrowania słabych bloków, który jest luźno wzorowany na algorytmie Tiny Encryption Algorithm (TEA). Słabsze odmiany TEA nazywane są Uproszczoną TEA (STEA). Sekcja 2 jest podstawowym wprowadzeniem do projektowania blokowego szyfru, podsumowującego rodzaje ataków, na które silny szyfr powinien być w stanie się oprzeć, opisując pojęcia zamętu i dyfuzji oraz wyjaśniając zasadę szyfrów Feistela. W sekcji 3 opisano szyfr blokowy TEA i wyjaśniono jego jedyną znaną słabość. W sekcji 4 wprowadzono szyfr blokowy STEA. Różne metody ataku na STEA, w tym bardzo skuteczny atak znanego ataku tekstem jawnym oraz niektóre metody kryptoanalizy kryptograficznej blokowej opisane są w sekcjach 5-10. Wreszcie, podziękowania znajdują się w sekcji 11.

Czynności wstępne

Ten tekst zakłada następującą notację dla niektórych operacji binarnych.

Exclusive-OR. Operacja dodawania n-krotek w polu $\mathbb{F}_2$ (znana również jako exclusive-or) jest oznaczona przez x $\oplus$ y.

Dodawanie całkowite Operacja dodawania liczby całkowitej modulo 2^n jest oznaczana przez x $\boxplus$ y (gdzie x, y $\in \mathbb{Z}_{2n}$). Wartość n powinna być jasna z kontekstu.

Odejmowanie całkowite Operacja odejmowania liczby całkowitej modulo 2^n jest oznaczana przez x $\boxminus$ y (gdzie x, y $\in \mathbb{Z}_{2n}$). Wartość n powinna być jasna z kontekstu. Zwróć też uwagę, że x $\boxminus$ y ≡ x$\boxplus$-y.

Przesunięcia bitowe Logiczne przesunięcie w lewo o x bitów y jest oznaczone przez x << y. Logiczne przesunięcie w prawo o x bitach y jest oznaczone przez x >> y.

Obroty bitowe Lewy obrót x o y bitów jest oznaczony przezx <<< y. Prawą rotację x o y bitów oznaczamy x >>> y.

Liczby szesnastkowe będą indeksowane przy pomocy ‘H’, np. 20_H = 32. Zbiór bloków tekstu jawnego oznaczony jest przez $\mathcal{P}$, zestaw bloków tekstu $\mathcal{C}$ a zbiór kluczy przez $\mathcal{K}$. Długość bloku komunikatu to m, gdzie | $\mathcal{P}$ | $\mathcal{C}$ | = 2^m, a długość klucza to k, gdzie | $\mathcal{K}$ | = 2^k.

2. Projekt bloku szyfrowego

Szyfr blokowy z kluczem symetrycznym E jest funkcją

$$E : \mathcal{P} \times \mathcal{K} \rightarrow \mathcal{C}. \qquad (1)$$

Co więcej, dla każdego klucza $K \in \mathcal{K}$ mamy funkcje

$$E_K : \mathcal{P} \rightarrow \mathcal{C},$$
$$D_K : \mathcal{C} \rightarrow \mathcal{P},$$

i $D = E^{-1}$. Innymi słowy, klucz determinuje bijektywne odwzorowanie tekstu jawnego na tekst zaszyfrowany; określa również odwrotne odwzorowanie zaszyfrowanego tekstu na tekst jawny. Jeśli klucz jest ustalony, a więc określa specyficzne mapowanie, szyfr blokowy może być po prostu traktowany jako duży odnośnik (szyfr zastępczy). W szczególności identyczne bloki zwykłego tekstu są szyfrowane do identycznych bloków tekstu zaszyfrowanego. Szyfrowanie zastępcze, w którym m jest małe (np. m = 8, co oznacza 256 odrębnych bloków komunikatów) jest niepewne, ponieważ jeśli atakujący ma wiedzę na temat wzorców statystycznych w zwykłym tekście, może próbować odzyskać tekst jawny z zaszyfrowanego tekstu za pomocą analizy częstotliwości . Innym problemem jest to, że jeśli atakujący poprawnie odgadnie tekst jawny, który odpowiada pewnemu zaszyfrowanemu tekstowi, wówczas może on zbudować tablicę przeglądowa par tekstu z tekstem jawnym - pary szyfrogramów odpowiadające określonemu kluczowi, znanemu jako książka kodowa. Do odszyfrowania bloków tekstu szyfrowanego bez znajomości klucza można użyć kodów. Oba problemy rozwiązuje się, zwiększając długość bloku, a tym samym zwiększając liczbę możliwych komunikatów. Atakujący nie może wykorzystać wiedzy o wzorcach tekstowych, które są krótsze niż długość bloku. Stwarza to również niepraktyczne dla osoby atakującej, która tworzy podręczniki kodowania tekstu z tekstem jawnym. Zazwyczaj długość bloku wiadomości powinna wynosić co najmniej 64 bity. Wymagamy również, aby był nieosiągalny dla atakującego, ze znanym tekstem jawnym (P; C), w celu odzyskania klucza poprzez wyczerpujące wyszukiwanie kluczy (brute-force). Oznacza to, że nie chcemy, aby ktokolwiek mógł określić klucz, próbując wartości $K \in \mathcal{K}$, dopóki nie znajdą klucza K takiego, że C = E (P; K). Przez wystarczająco dużą długość klucza możemy uczynić ataki brutalną siłą obliczeniowo niewykonalne (szyfr będzie wówczas bezpieczny obliczeniowo). Długość klucza k powinna wynosić co najmniej 64 bity1. Decydując o parametrach długości bloku i długości klucza, projektant musi rozwinąć szyfr blokowy w taki sposób, aby mógł on pokonać lub oprzeć się następującym typom ataków, gdzie osoba atakująca chciałaby odzyskać klucz:

Tylko tekst zaszyfrowany. Osoba atakująca ma wiedzę o pewnym zaszyfrowanym tekście, ale nie o zwykłym tekście ani kluczu. W takim przypadku odzyskanie jawnego tekstu (bez klucza) może być udanym atakiem.

Znany tekst jawny Osoba atakująca ma wiedzę na temat jakiegoś zaszyfrowanego tekstu i odpowiedniego tekstu jawnego, zaszyfrowanego pod nieznanym kluczem

Wybrany tekst jawny Osoba atakująca ma możliwość wybrania dowolnych wiadomości do zaszyfrowania pod nieznanym kluczem.

Adaptacyjny wybrany tekst jawny Osoba atakująca ma nieograniczoną możliwość posiadania wiadomości, które wybrał zaszyfrowanie pod nieznanym kluczem. Spodziewamy się, że

adaptacyjny wybrany atak zwykłego tekstu wymagałby mniejszej liczby zaszyfrowań niż prawdziwy wybrany atak tekstowy.

Projektant podąża za założeniem Kerckhoffa , że atakujący ma pełną wiedzę na temat działania systemu szyfrowania. Zabezpieczenie szyfru powinno spoczywać w całości na kluczu.

2.1. Zamieszanie i dyfuzja

Z wyjątkiem małych bloków komunikatów lub długości kluczy nie jest możliwe, aby projektant szyfrów blokowych jawnie określał tekst jawny do odwzorowania zaszyfrowanego tekstu dla każdego możliwego klucza; to byłoby jak wyszczególnianie 2^k książek kodowych. O wiele bardziej praktyczne jest określenie szyfru blokowego jako równania lub algorytmu. Na przykład prosty szyfr blokowy z parametrami (m; k) = (64,64) może być określony przez

$$\begin{aligned} C &= P \oplus K, \\ P &= C \oplus K. \end{aligned} \qquad (2)$$

Problem z szyfrem blokowym równania 2 polega na tym, że jest on trywialnie łamany jednym znanym tekstem jawnym przez

$$K = P \oplus C.$$

Ten blokowy szyfr jest słaby, ponieważ jest czysto liniowy, a więc łatwo rozwiązywany. Korzystając zarówno z operacji liniowych, jak i nieliniowych, szyfru blokowego jest trudniej manipulować prostą algebrą. Na przykład nieco lepszym pomysłem byłby szyfr blokowy (m, k) = (64, 128),

$$\begin{aligned} C &= (P \oplus K_0) \boxplus K_1, \\ P &= (C \boxminus K_1) \oplus K_0, \end{aligned} \qquad (3)$$

gdzie K_0 i K_1 są podkluczami, tj. zmiennymi zależnymi od klucza. W tym przypadku każdy podklucz stanowi połowę klucza K, a zatem ma długość 64-bitową. Aby odzyskać klucz K, atakujący musiałby rozwiązać równanie 3. Ponieważ istnieją dwie niewiadome, unikalne rozwiązanie nie zostałoby znalezione tylko z jednym znanym tekstem jawnym. Jednak z dwoma znanymi tekstami jawnymi

, (P,C) i (P' , C'), mamy równania

$$\begin{aligned} C &= (P \oplus K_0) \boxplus K_1, \\ C' &= (P' \oplus K_0) \boxplus K_1. \end{aligned}$$

Atakujący może odjąć (modulo 2^{32}) jeden szyfrogram od siebie, uzyskując

$$C \boxminus C' = (P \oplus K_0) \boxminus (P' \oplus K_0), \qquad (4)$$

gdzie 64-bitowe hasło podklucza K_1 zostało wyeliminowane. Jednakże, równanie 4 nie może być dalej manipulowane do równania pozostałego podklucza K_0 pod względem bloków tekstu jawnego i bloków szyfrogramu, ponieważ operacje $\oplus$ i $\boxplus$ nie postępuj zgodnie z przepisami dotyczącymi dystrybucji. Co więcej, nie ma prostej algebraicznej zależności między dodawaniem modulo 2^m a wyłącznymi lub of-bitowymi wektorami, ponieważ grupy ($\mathbb{Z}_{2m}$, $\boxplus$) i ($\mathbb{F}_2{}^m$, $\oplus$) nie są izomorficzne]. Idea mieszania operacji liniowych i nieliniowych w celu zaciemnienia relacji między tekstem jawnym, szyfrogramem i

kluczem, nazywana jest zamętem i jest ważną zasadą projektowania szyfru. Równie ważną zasadą projektowania szyfru blokowego jest dyfuzja, tzn. Idea, że każdy fragment tekstu zaszyfrowanego powinien zależeć od każdego fragmentu zwykłego tekstu i każdego fragmentu klawisza . Zapewnia to rozproszenie statystyk tekstu jawnego w zaszyfrowanym tekście, tak aby osoba atakująca nie mogła przewidzieć tekstu jawnego, który odpowiada konkretnemu zaszyfrowanemu tekstowi, nawet po zaobserwowaniu wielu "podobnych" tekstów jawnych i odpowiadających im zaszyfrowanych tekstów. Prostą metodą uzyskania pomyłki i dyfuzji w szyfrze blokowym jest wielokrotne stosowanie podstawionych kluczy i permutacji do wiadomości. Podstawienia są używane do wprowadzenia nieliniowości do komunikatu (dezorientacja), a permutacje są wymagane w celu zapewnienia, że na bity wpływają różne substytucje w kolejnych iteracjach (dyfuzja). Szyfr blokowy oparty na tej zasadzie nazywany jest iterowanym szyfrem. W praktyce projektant opracowuje funkcję ρ zwaną funkcją okrągłą, która ma wejścia bloku wiadomości X i podklucza K i wyprowadza częściowo zaszyfrowany blok wiadomości Y = ρ(X; K). Funkcja okrągła jest odpowiedzialna za spełnienie podstawowych wymagań dotyczących pomyłki i dyfuzji, dlatego chcielibyśmy, aby każdy bit X i każdy bit K wpływał na każdy bit Y w sposób nieliniowy, ale odwracalny. Na przykład funkcja okrągła zdefiniowana przez

$$\rho(X, K) \stackrel{\text{def}}{=} (X \boxplus K) \lll 3 \qquad (5)$$

spełnia wymóg dotyczący pomyłki poprzez mieszanie nieliniowego dodawania liczby całkowitej z liniową permutacją bitową i wymagania dyfuzyjne przez permutację bitową. Zauważ, że funkcja ta sama w sobie jest nieodpowiednia jako szyfr blokowy, ponieważ jest łatwa do rozwiązania. Przez powtarzanie funkcji okrągłej ustaloną liczbę razy automatycznie uzyskujemy pewne zabezpieczenia jako konsekwencja faktu, że po każdej iteracji (lub rundzie) bity wyjściowe stają się bardziej zależne od bitów wejściowych. Możemy użyć różnych podkluczy w każdej iteracji, aby dla 4-okrągłych bloków szyfrów funkcja szyfrowania stała się

$$C = E(P, K) = \rho(\rho(\rho(\rho(P, K_1), K_2), K_3), K_4).$$

Podobnie, odszyfrowanie mogłoby zostać osiągnięte przez

$$P = D(C, K) = \rho^{-1}(\rho^{-1}(\rho^{-1}(\rho^{-1}(C, K_4), K_3), K_2), K_1)$$

Podklucze pochodziłyby z klucza za pomocą algorytmu kluczowania harmonogramu (który również musiałby zostać zaprojektowany ostrożnie). Ustalenie liczby rund wymaganych dla odpowiedniego poziomu bezpieczeństwa kryptograficznego nie jest łatwe. To zależy od projektu i bezpieczeństwa zapewnianego przez funkcję okrągłą. Wybierając dużą liczbę rund, które pokonają najbardziej nowoczesne ataki kryptoanalityczne, może to również spowodować, że szyfr będzie zbyt powolny z wielu praktycznych powodów. Projektanci szyfrów blokowych określają liczbę rund po zbadaniu podatności szyfru na różne ataki ogólne. Zazwyczaj wybiera się "konserwatywną" liczbę rund, która jest o kilka rund większa niż minimalna wymagana liczba rund (tj. liczba rund, które spowodowałyby, że szyfr padłby na pewien rodzaj ataku). Potrzeba oddzielnej funkcji odszyfrowywania można uniknąć przez zaprojektowanie funkcji zaokrąglenia jako inwolucji (tj. $\rho^{-1} = \rho$). Aby odszyfrować, funkcja szyfrowania byłaby zastosowana do tekstu zaszyfrowanego, ale z odwróconymi podkluczami (w odniesieniu do jakiejkolwiek operacji jest używana do połączenia ich z danymi) i odwrócona kolejność podkluczy. Skutecznie,

$$P = D(C, K) = \rho(\rho(\rho(\rho(C, K_4^{-1}), K_3^{-1}), K_2^{-1}), K_1^{-1})$$

Może to wiązać się z dodatkowymi pracują przy obliczaniu podkluczy odszyfrowania z podkluczy szyfrowania (jak w przypadku szyfru IDEA , ale będzie to obsługiwane przez algorytm klucza harmonogramu i będzie przejrzyste dla funkcji szyfrowania. Szyfr, który ma właściwość to nazywa się E/D podobny. Jednym z problemów spowodowanych tą strukturą szyfrów jest możliwość jednokrotnego zaszyfrowania wiadomości, a następnie jej jednokrotne odszyfrowanie po prostu poprzez wybranie złych podkluczy. W szczególności runda może anulować efekt z poprzedniej rundy, jeśli jej podklucz jest odwrotnością poprzedniego podklucza rundy (tj., Jeśli $K_i = K^{-1}_{i-1}$ dla rundy i). W takim przypadku dwie rundy zostają zmarnowane (nie będą miały wpływu na blok komunikatów), a więc ta słabość może obniżyć ogólne bezpieczeństwo szyfru blokowego. Istnieją dwa sposoby korygowania tego projektu. Po pierwsze, projektując kluczowy harmonogram, który gwarantuje, że każdy podklucz rundy nigdy nie jest odwrotnością poprzedniego podklucza rundy. Po drugie, komponując okrągłą funkcję fuzji (aby zachować podobieństwo E / D), ale zapewniając, że sama funkcja rundy nie jest inwolucji. Druga opcja jest zwykle lepsza od pierwszej szyfr powinien być dość silny, niezależnie od głównego harmonogramu). Szyfry podobne do E / D są "symetryczne" w swojej konstrukcji i możemy korzystać z wygodnej notacji zilustrować ten punkt, nadając funkcji ? ρ indeks oznaczający podklucz używany w danej rundzie. Na przykład,

$$\begin{aligned} C = E(P,K) &= \rho_1(\rho_2(\rho_3(\rho_4(P)))), \\ P = D(C,K) &= \rho_4(\rho_3(\rho_2(\rho_1(C)))). \end{aligned}$$

Jeśli funkcja rundy nie jest inwolucją, to po rundzie finałowej następowałaby transformacja wyjściowa, której celem jest spowodowanie, by szyfr był symetryczny (a więc podobny do E / D). Wiele współczesnych szyfrów blokowych wykorzystuje tę samą "ogólną" okrągłą funkcję co szyfr DES, ponieważ ma użyteczną właściwość, że funkcja szyfrowania jest podobna do E / D. Są one znane jako szyfry Feistela.

2.2 Szyfry Feistela

Podstawową zasadą szyfrów Feistela jest to, że blok tekstu jawnego jest podzielony na dwie połowy, a każda połowa służy do szyfrowania drugiej połowy przez określoną liczbę rund. Wynikiem jest blok tekstu zaszyfrowanego złożony z dwóch zaszyfrowanych półbloków. Funkcja okrągła składa się z przymusowego włączenia i permutacji, która działa na dwóch blokach, każdy o długości m/2. Część podstawienia funkcji okrągłej, którą oznaczamy przez σ, jest zdefiniowana przez

$$\sigma_i(L,R) = (L \oplus F(R,K_i), R).$$

gdzie F jest funkcją szyfrowania, a K_i jest podkluczem. Możemy pokazać że σ jest inwolucją, pokazując, że $\sigma^2(L; R) = (L,R)$, jak następuje

$$\begin{aligned} \sigma_i^2(L,R) &= \sigma_i(L \oplus F(R,K_i), R) \\ &= (L \oplus F(R,K_i) \oplus F(R,K_i), R) \\ &= (L,R). \end{aligned}$$

Pokazuje to również, że funkcja szyfrowania nie musi być odwracalna. Część permutacyjna funkcji okrągłej, którą oznaczamy przez π, jest zdefiniowana przez π(L,R) = (R,L); co jest prostą "zamianą" połówek. Jest to oczywiście inwolucji. Naszym celem jest stworzenie podobnego szyfru E / D na przemian z zastosowaniem substytucji σ i permutacji π. Zarówno σ jak i π są inwolucjami, można to łatwo osiągnąć, zapewniając, że szyfr kończy się z tą samą funkcją, która jest stosowana najpierw (stąd

uczynienie szyfru "symetrycznym"). Zaokrąglona funkcja szyfru Feistela to ρ(L, R) = π(σ(L, R)). To nie jest inwolucji, ale szyfr jest E / D podobny, jeśli runda zastąpienia σ jest pierwszym i ostatnim etapem szyfru:

$$\begin{aligned} C = E(P,K) &= \sigma_n(\rho_{n-1}(\cdots(\rho_1(P))\cdots)), \\ P = D(C,K) &= \sigma_1(\rho_2(\cdots(\rho_n(C))\cdots)). \end{aligned}$$

W praktyce funkcja okrągła ρ stosuje się n razy, a następnie transformację wyjściową π jest zastosowany do cofnięcia π permutacja w n-tej rundzie (więc ostatnia efektywna operacja na bloku wiadomości będzie częścią zastępującą σ_n, w razie potrzeby). Kolejną kwestią, którą musimy rozwiązać, jest to, w jaki sposób podklucze do deszyfrowania pochodzą z podkluczy szyfrowania. Wspomniano, że podklucze deszyfrowania będą odwrotnością operacji, która jest używana do łączenia ich z danymi. W funkcji?, Zależne od klucza wyjście funkcji szyfrowania jest połączone z blokiem wiadomości za pomocą operacji exclusive-or. Dlatego podklucze odszyfrowywania są takie same jak podklucze szyfrowania, ponieważ odwrotność dowolnego elementu, z odniesieniu do ⊕, samo w sobie. Jedyną różnicą między szyfrowaniem a odszyfrowywaniem jest kolejność, w której stosowane są podklucze, stąd ta konstrukcja jest podobna do E/D. Dokładniej, tekst jawny jest podzielony na dwa bloki, P = (L_0 , R_0). Następnie, dla szyfru n-rundowego, iteruje się następującą okrągłą funkcję (dla 1 ≤ i ≤ n):

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus F(R_{i-1}, K_i), \end{aligned} \qquad (6)$$

gdzie F jest funkcją szyfrowania, a K_i są podkluczami. Tekst zaszyfrowany to C = (R_n, L_n). Zwróć uwagę, że kolejność połówek wiadomości jest odwrócona na wyjściu; wynika to z transformacji wyjścia (która cofa zamianę n-tej rundy). Ogólnie możemy użyć dowolnej operacji grupowej ⊗ zamiast ⊕ W równaniu 6, aby funkcja okrągła mogła być

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \otimes F(R_{i-1}, K_i). \end{aligned}$$

Funkcją okrągła wymagana do odszyfrowania będzie

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \otimes (F(R_{i-1}, K_i))^{-1}, \end{aligned}$$

gdzie konieczne byłoby obliczenie odwrotności (w odniesieniu do operacji ⊗) funkcji wyjścia szyfrui. W związku z tym runda odszyfrowywania może nie być taka sama jak runda szyfrowania, w którym to przypadku szyfr nie będzie już podobny do E/D. Przykładami szyfrów wykorzystujących tę zmodyfikowaną strukturę Feistela są szyfry blokowe TEA i STEA, które wykorzystują całkowite dodawanie modulo 2^{32} (operacja ⊞) Zamiast wyłącznej - lub w konsekwencji zarówno TEA jak i STEA wymagają oddzielnych funkcji deszyfrowania. Jedynymi częściami szyfru blokowego, które nie są określone przez strukturę Feistela, są: konstrukcja funkcji szyfrowania i algorytm harmonogramu klucza, który określa, w jaki sposób podklucze (K1;…,K_n) pochodzą z klucza K.

3 Szyfr blokowy TEA

Algorytm blokowy TEA (Tiny Encryption Algorithm) został zaprezentowany na warsztatach Fast Software Encryption w 1994 roku [15]. Jest to 64-stopniowy szyfr Feistela działający na 64-bitowych

blokach wiadomości z kluczem 128-bitowym. Został zaprojektowany do implementacji oprogramowania i wszystkie jego operacje są oparte na 32-bitowych słowach i używają operacji arytmetycznych i logicznych, zamiast używania (tradycyjnych) podstawień i tabel permutacji w funkcji szyfrowania. Jak wspomniano wcześniej, funkcja okrągła różni się nieco od zwykłej specyfikacji tym, że dodanie liczby całkowitej jest modulo 232 zamiast wyłącznego - lub jako operator łączący:

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \boxplus F(R_{i-1}, K_i), \end{aligned} \qquad (7)$$

Nie ma końcowego odrzutu, ale to nie jest problem, ponieważ funkcja odszyfrowywania jest osobna i zajmuje się tym. Tekst jawny to P = (L_0,R_0), a tekst zaszyfrowany to C = (L_{64},, R_{64}). Każdy podklucz K_i jest uporządkowaną 3-krotną liczbą elementów w Z_{232}, postaci (T, U , V), gdzie (T , U) są 64 bitami tajnego klucza, a V jest znaną stałą 32-bitową. Funkcja szyfrowania F jest zdefiniowana przez

$$F(M,(T,U,V)) \stackrel{\text{def}}{=} ((M \ll 4) \boxplus T) \oplus ((M \gg 5) \boxplus U) \oplus (M \boxplus V). \qquad (8)$$

Aby wygenerować podklucze, najpierw 128-bitowy klucz K jest podzielony na cztery 32-bitowe bloki K = (W, X, Y, Z), a wtedy podklucze K_i są określone przez

$$\begin{aligned} \delta_i &= \delta \left\lfloor \frac{i+1}{2} \right\rfloor, \\ K_i &= \begin{cases} (W, X, \delta_i) & \text{jeśli i jest nieparzyste} \\ (Y, Z, \delta_i) & \text{jeśli i jest parzyste} \end{cases} \quad (\text{dla } 1 \le i \le 64). \end{aligned}$$

Dla każdego klucza K podklucze Ki są różne z powodu alternatywnych podkluczy mających różne wielokrotności? według głównego harmonogramu. Kluczowa stała harmonogramu δ = $9E3779B9_H$ pochodzi od złotej liczby, ale jej dokładna wartość nie ma znaczenia kryptograficznego; jej głównym celem jest zapewnienie, że podklucze są odrębne. Funkcja rundy TEA jest znacznie bardziej skomplikowana niż proste szyki blokowe w poprzednich sekcjach tego raportu, co może wywołać następujące pytania:

1. Dlaczego stosuje się 64 rundy?

2. Dlaczego kluczowy harmonogram jest stały? wymagany?

3. Dlaczego blok wiadomości jest przesunięty w funkcji szyfrowania?

4. Czy są jakieś słabości w tym systemie?

Na ostatnie pytanie można odpowiedzieć od razu, ale odpowiedzi na pierwsze trzy pytania powinny być widoczne w późniejszych rozdziałach, kiedy analizujemy szyfr STEA.

3.1 Równoważne klucze

Dla każdego systemu szyfrowania klucz K' jest równoważny kluczowi K wtedy i tylko wtedy, gdy

$E_K(P) = E_{K'}(P)$: (9)

Ponieważ jest to relacja równoważności, możemy zdefiniować klasy równoważności w K i powiedzieć, że para kluczy K i K' należą do tej samej klasy równoważności, jeśli zachowane jest równanie 9. W przypadku dobrze zaprojektowanego szyfru nie chcielibyśmy, aby istniały równoważne klucze (każdy

klucz K ∈ $\mathcal{K}$ powinien określać wyraźne odwzorowanie tekstu jawnego na tekst zaszyfrowany), a więc oczekiwalibyśmy równoważności klas 2^k za pomocą równania. Tak więc dla szyfrów TEA można oczekiwać 2^{128} klas równoważności. W rzeczywistości jednak TEA ma 2^{126} klas równoważności (2^{126} różnych odwzorowań tekstu jawnego na szyfrogram). Ta słabość przejawia się w funkcji szyfrującej (zwykle najbardziej wrażliwej części szyfru Feistela). Zwróć uwagę, że dla wszystkich wartości X, Y ∈ $\mathbb{Z}_{232}$,

$$\begin{aligned} 2^{31} \boxplus 2^{31} &= 0, \\ X \boxplus 2^{31} &= X \oplus 80000000_{\mathrm{H}}. \end{aligned}$$

Zatem

$$X \boxplus Y \equiv (X \oplus 80000000_{\mathrm{H}}) \boxplus (Y \oplus 80000000_{\mathrm{H}})$$

Funkcję szyfrowania (równanie 8) można w podobny sposób udowodnić

$$F(M,(T,U,V)) \equiv F(M,(T \oplus 80000000_{\mathrm{H}}, U \oplus 80000000_{\mathrm{H}}, V))$$

Oznacza to, że każdy 128-bitowy klucz K = (W, X, Y, Z) ma trzy równoważne klucze w formularzu:

$$\begin{gathered} (W \oplus 80000000_{\mathrm{H}}, X \oplus 80000000_{\mathrm{H}}, Y, Z), \\ (W, X, Y \oplus 80000000_{\mathrm{H}}, Z \oplus 80000000_{\mathrm{H}}), \\ (W \oplus 80000000_{\mathrm{H}}, X \oplus 80000000_{\mathrm{H}}, Y \oplus 80000000_{\mathrm{H}}, Z \oplus 80000000_{\mathrm{H}}). \end{gathered}$$

Na przykład zaszyfrowanie zwykłego tekstu P = (00000000 00000000_{H}) za pomocą któregokolwiek z poniższych kluczy powoduje wygenerowanie zaszyfrowanego tekstu C = (9327C497 $31B08BBE_{\mathrm{H}}$):

$$\begin{aligned} &00000000\ 80000000\ 00000000\ 00000000_{\mathrm{H}}, \\ &80000000\ 00000000\ 00000000\ 00000000_{\mathrm{H}}, \\ &80000000\ 00000000\ 80000000\ 80000000_{\mathrm{H}}, \\ &00000000\ 80000000\ 80000000\ 80000000_{\mathrm{H}}. \end{aligned}$$

Ta słabość oznacza, że chociaż TEA używa klucza 128-bitowego, zapewnia w najlepszym przypadku to samo bezpieczeństwo co klucz 126-bitowy. W wyczerpującym wyszukiwaniu kluczy osoba atakująca musiałaby przetestować tylko jedną czwartą całkowitej liczby możliwych kluczy (ponieważ nie ma potrzeby wypróbowywania kluczy równoważnych z tymi, które zostały już przetestowane). Pomimo równoważnych kluczy, zmniejszenie złożoności wyczerpującego wyszukiwania kluczy nie jest zagrożeniem dla bezpieczeństwa TEA (chociaż uwydatnia to wadę projektu), ponieważ nadal jest bezpieczna obliczeniowo. Jednak istnienie równoważnych kluczy oznacza, że TEA nie nadaje się do użycia w funkcji mieszającej, która jest oparta na szyfrach blokowych. Aby wyeliminować równoważne klucze, konieczne byłoby przeprojektowanie harmonogramu kluczy TEA, co właśnie zrobili projektanci TEA. Ich ulepszenia również pokonują "powiązany klucz" ataku na TEA, który wymaga 2^{23} wybranych tekstów jawnych zaszyfrowanych pod dwoma powiązanymi kluczami (wartość drugiego klucza zależy od wartości pierwszego klucza)

4 Szyfr blokowy STEA

Szyfr blokowy STEA to 64-stopniowy szyfr Feistela działający na 64-bitowych blokach komunikatów z kluczem 64-bitowym. Został zaprojektowany wyłącznie jako blok testowy do nauki o nowoczesnych współczesnych technikach kryptoanalitycznych i istnieje wiele odmian tego samego celu (ale zaprojektowanych w celu zilustrowania różnych ataków). Szyfr blokowy STEA wykorzystuje funkcję okrągłą jak w równaniu 7 i funkcję szyfrowania zdefiniowaną przez

$$F(M, K_i) = M \oplus K_i, \tag{10}$$

gdzie K_i jest 32-bitowym podkluczem. Biorąc pod uwagę klucz 64-bitowy jako dwa 32-bitowe bloki, tak że $K = (K_1, K_2)$, podklucze to po prostu $K_i = K_1$ dla i nieparzystego, a $K_i = K_2$ dla i parzystego (gdzie $1 \leq i \leq 64$). Tak więc połówki klucza są stosowane w alternatywnych rundach. Aby zaszyfrować, tekst jawny $P = (L_0, R_0)$ jest wprowadzany do funkcji szyfrowania, która iteruje funkcję okrągłą

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \boxplus (R_{i-1} \oplus K_i), \end{aligned} \tag{11}$$

$1 \leq i \leq 64$

Wynik jest tekstem zaszyfrowanym $C = (L_{64}, R_{64})$. Do odszyfrowania, tekst zaszyfrowany $C = (L_0, R_0)$ jest wprowadzany do funkcji deszyfrowania, która iteruje

$$\begin{aligned} R_i &= L_{i-1}, \\ L_i &= R_{i-1} \boxminus (L_{i-1} \oplus K_i), \end{aligned}$$

dla $1 \leq i \leq 64$. Tekst jawny to $P = (L_{64}, R_{64})$. Podklucze musiałyby być stosowane w odwrotnej kolejności, jak zwykle, a więc $K_i = K_2$ dla i nieparzystych, a $K_i = K_1$ dla i parzystych (gdzie $1 \leq i \leq 64$). Aby pokazać efektywność STEA, poniższa tabela wymienia niektóre jawne teksty (wybrane jako niskie wagi Hamminga i odległości od innych) i odpowiednie zaszyfrowane teksty:

Klucz	Tekst jawny	Tekst zaszyfrowany
$(00000000\ 00000000_H)$	$(00000000\ 00000000_H)$	$(00000000\ 00000000_H)$
$(00000000\ 00000000_H)$	$(00000000\ 00000001_H)$	$(61CA20BB\ 297A859D_H)$
$(00000000\ 00000000_H)$	$(00000001\ 00000001_H)$	$(297A859D\ 8B44A658_H)$
$(00000000\ 00000000_H)$	$(00000001\ 00000000_H)$	$(C7B064E2\ 61CA20BB_H)$
$(00000001\ 00000003_H)$	$(00000000\ 00000000_H)$	$(57112EA4\ 255E6231_H)$
$(00000001\ 00000003_H)$	$(00000000\ 00000001_H)$	$(F12AEA7D\ ED0EC712_H)$
$(00000001\ 00000003_H)$	$(00000001\ 00000001_H)$	$(C7B064E3\ 61CA20BB_H)\star$
$(00000001\ 00000003_H)$	$(00000001\ 00000000_H)$	$(C7B064E2\ 61CA20BA_H)\star$
$(55555555\ 55555555_H)$	$(22222222\ 22222222_H)$	$(0F3102B4\ 83B32497_H)$

Dwa bloki tekstu zaszyfrowanego oznaczone symbolem * są bardzo podobne, różniące się dokładnie w miejscach, w których ich korespondujące teksty jawne różnią się. Wskazuje to na poważny problem spowodowany brakiem dyfuzji. W powyższej tabeli istnieją inne zauważalne wzorce, ale z punktu ataku tylko zaszyfrowanego nie wydaje się, aby istniało jakiekolwiek podobieństwo między tekstami jawnymi a ich odpowiednimi zaszyfrowanymi tekstami (z wyjątkiem przypadku, w którym tekst jawny i klucz są zerowe; Powód tego zostanie wyjaśniony później). STEA nie dziedziczy równorzędnego problemu

kluczy, na które cierpi TEA. Ma jednak szereg niedociągnięć ze względu na zbyt uproszczoną funkcję szyfrowania i kluczowy harmonogram.

5 Rozwiązanie algebraiczne

Jak wyjaśniono w sekcji 2, funkcja rundy nie musi być zabezpieczona przed rozwiązaniem algebraicznym, aby szyfr był bezpieczny, pod warunkiem, że używana jest wystarczająca liczba rund. W 2-rundowej STEA, tekst jawny P = (L_0,R_0) i tekst zaszyfrowany C = (L_2 , R_2) są powiązane przez

$$\begin{aligned} L_2 &= L_0 \boxplus (R_0 \oplus K_1), \\ R_2 &= R_0 \boxplus (L_2 \oplus K_2). \end{aligned}$$

W każdym równaniu jest dokładnie jedna niewiadoma. Zatem STEA potrzebuje co najmniej trzech rund, w przeciwnym razie możemy odzyskać klucz 64-bitowy z tylko jednym znanym tekstem jawnym przez

$$\begin{aligned} K_1 &= (L_2 \boxminus L_0) \oplus R_0, \\ K_2 &= (R_2 \boxminus R_0) \oplus L_0. \end{aligned}$$

W rzeczywistości każdy szyfr Feistela wymagałby co najmniej trzech rund, ponieważ część tego problemu spowodowana jest faktem, że każda runda szyfru Feistela szyfruje tylko blok o połowie wiadomości. Jeśli podklucz można odzyskać z wejścia i wyjścia funkcji szyfrowania (jak w STEA), klucz można odzyskać tylko jednym znanym tekstem jawnym.

6 Atak Meet-in-the-Middle

Jeśli spojrzymy na 3-rundową STEA z niezależnymi podkluczami, tj. bez harmonogramu klucza (tak, że kluczowa długość efektywnie staje się k = 3 x 32 = 96), stwierdzamy, że tekst jawny P = (L_0, R_0) i szyfrogram C = (L_3, R_3) są spokrewnione przez

$$\begin{aligned} L_3 &= R_0 \boxplus ((L_0 \boxplus (R_0 \oplus K_1)) \oplus K_2), \\ R_3 &= L_0 \boxplus (R_0 \oplus K_1) \boxplus (L_3 \oplus K_3). \end{aligned} \tag{12}$$

W ataku znanym tekstem jawnym równanie 12 dzieli cztery znane terminy (każda połowa tekstu jawnego i tekstu zaszyfrowanego) i trzy nieznane stałe (trzy podklucze). Nie ma sposobu na odzyskanie żadnego z podkluczy po prostu przez manipulowanie tymi równaniami. Wyczerpujące wyszukiwanie kluczy w tym wariancie STEA wymagałoby 2^{96} szyfrowania i jednego znanego tekstu jawnego. Atak typu "meet-in-the middle" znajduje użyteczny kompromis pomiędzy pamięcią i obliczeniami, w celu efektywnego wyszukiwania kluczy. Równanie 12 można przestawić na

$$\begin{aligned} K_2 &= (L_3 \boxminus R_0) \oplus (L_0 \boxplus (R_0 \oplus K_1)), \\ K_3 &= L_3 \oplus (R_3 \boxminus L_0 \boxminus (R_0 \oplus K_1)). \end{aligned} \tag{13}$$

W ataku znanym tekstem jawnym, równanie 13 ma podklucze K_2 i K_3 wyrażone w terminach czterech znanych terminów i jednej nieznanej stałej 32-bitowej (podklucz K_1). Z jednym znanym tekstem jawnym i próbując wszystkich wartości $K_1 \in \mathbb{F}^{32}_2$, możemy znaleźć odpowiednie wartości K_2 i K_3. W efekcie będziemy mieli 2^{32} możliwe wartości dla klucza 96-bitowego. Kolejny znany blok tekstu jawnego jest następnie szyfrowany za pomocą każdego z tych 2^{32} możliwych kluczy; jeśli wynik

szyfrowania próbnego pokrywa się z rzeczywistym tekstem zaszyfrowanym, to z dużym prawdopodobieństwem odzyskaliśmy klucz. Tak więc, aby odzyskać 3-okrągły 96-bitowy klucz STEA w ataku "spotkać w środku", wymagane jest co najwyżej 233 szyfrowania i dwa znane zwykłe teksty. Atak typu "spotkanie w środku" na 4-rundzie STEA z niezależnymi podkluczami (k = 128) wymaga co najwyżej 2^{65} szyfrowań i dwóch znanych tekstów jawnych. Przy harmonogramie kluczy STEA (k = 64) atak na cztery rundy jest mniej wydajny niż wyczerpujące wyszukiwanie kluczy. Atak typu "meet-in-the-midle" można również zastosować do szyfru blokowego równania 3. Zauważ, że

$$K_1 = C \boxminus (P \oplus K_0)$$

Teraz możemy odzyskać klucz używając dwóch znanych jawnych tekstów w następujący sposób. Dla każdej pary znanych tekstów spróbuj użyć każdej możliwej wartości $K_0 \in \mathbb{F}^{64}{}_2$ i uzyskaj dwie możliwe wartości K_1. Jeśli dwie możliwe wartości K1 są zgodne, to znaleźliśmy właściwy klucz. Oznacza to, że musimy spróbować kodowania przy 2^{65}, aby odzyskać klucz, w porównaniu z 2^{128}, który byłby wymagany do wyczerpującego wyszukiwania kluczy. Istnieje wiele odmian implementacji ataku meet-in-the-middle. Naiwną próbą podwojenia efektywnej długości klucza szyfru blokowego może być zaszyfrowanie zwykłego tekstu za pomocą klucza K_1, a następnie odszyfrowanie wyniku za pomocą innego klucza K_2, tak aby

C = D (E (P, K1), K_2)

wyszukiwanie wyczerpującym kluczem wymagałoby do 2^{2k} prób dla klucza (K_1, K_2) i jednego znanego jawnego tekstu. Jednak atak typu "spotkać w środku" może znacznie szybciej odzyskać klucz. Dla k ≤ m, atak można wykonać w następujący sposób. Niech (P , C) i (P’, C’) będą dwoma znanymi tekstami jawnymi. Używamy tabeli kontrolnej T i oznaczamy i-ty wpis w T przez T(i). Uzupełnij tabelę odnośników o T (E(P, K_1)) = K_1, dla każdego klucza $K_1 \in \mathbb{F}^2{}_k$. Teraz dla każdego klucza K_2, niech K_1 = T(E (C,K_2)) i oblicz kodowanie próbne P’; jeśli C’ = D(E (P’, K_1), K_2), to znaleźliśmy poprawny klucz 2k-bitowy (K_1, K_2). Ten atak typu "meet-in-the middle" odzyskuje klucz z próbami do 2^{k+1}, pamięcią dla wiadomości 2^k i dwoma znanymi tekstami jawnymi. Jeśli k > m wtedy struktura tablicy odnośników musi być nieco inna, ponieważ każda T(i) może mieć więcej niż jedną wartość. Ataki typu "meet-in-the middle" są przydatne tylko wtedy, gdy możemy rozłożyć proces szyfrowania na więcej niż jedno równanie do rozwiązania, każde równanie ma inne nieznane terminy, a całkowita długość nieznanych terminów w każdym równaniu jest mniejsza niż klucz. długość k. Jeśli w szyfrze Feistela stosuje się więcej niż trzy rundy, wówczas atak typu "kompozycja" w środkowej części staje się niepraktyczny, ponieważ złożoność ataku rośnie wykładniczo wraz z długością każdego podklucza dla każdej dodatkowej rundy.

7 Atak Dziel i Rządź

Istnieje bardzo skuteczny atak znanym tekstem jawnym na STEA, który może odzyskać cały 64-bitowy klucz z tylko 32 kodami. Atak ten opiera się na zasadzie dziel i rządź, w której struktura szyfrów jest podzielona na części, a mniejsze części są atakowane oddzielnie. Do tej pory dokonaliśmy przeglądu STEA tylko z ogólnego punktu algebraicznego. Teraz przyjrzymy się bliżej procesowi zamętu i dyfuzji i ujawnił słabość słabości STEA. Oznaczamy i-ty bit A przez A [i], gdzie najmniej znaczącym bitem (LSB) z A jest A [0]. Algorytm obliczania S = A ⊞ B obejmuje równania

$$\begin{aligned} C[0] &= 0, \\ S[i] &= A[i] \oplus B[i] \oplus C[i], \\ C[i+1] &= A[i]B[i] \oplus A[i]C[i] \oplus B[i]C[i]. \end{aligned} \tag{14}$$

gdzie S oznacza sumę, a C oznacza wartość przeniesienia. Pierwsza uwaga, że S[0] = A[0] ⊕ B[0]; LSB jest liniowe w stosunku do $\mathbb{F}_2$. Po drugie, jedyna nieliniowość wynikająca z dodawania wynika z przenoszenia, które rozprzestrzenia się tylko w górę (dyfuzja w jednym kierunku). Funkcja zaokrąglania STEA dla najmniej znaczącego bitu bloku wiadomości to liniowy

$$\begin{aligned} L_i[0] &= R_{i-1}[0] \\ R_i[0] &= L_{i-1}[0] \oplus R_{i-1}[0] \oplus K_i[0] \end{aligned}$$

W ten sposób możemy odzyskać dwa kluczowe bity (najmniej znaczące bity K_1 i K_2) przy użyciu znanego zwykłego tekstu poprzez znalezienie wyrażeń liniowych dotyczących LSBs tekstu jawnego, szyfrogramu i klucza. Oznaczając tekst jawny P = (L_0 , R_0) i tekst zaszyfrowany C = (L_{64} , R_{64}), otrzymujemy następujące równania po 64 rundach:

$$\begin{aligned} K_1[0] &= L_{64}[0] \oplus R_{64}[0] \oplus L_0[0], \\ K_2[0] &= L_0[0] \oplus R_0[0] \oplus R_{64}[0]. \end{aligned} \tag{15}$$

Terminy po prawej stronie są znane, a zatem odzyskujemy wartości $K_1[0]$ i $K_2[0]$. Ponieważ każdy bit komunikatu jest niezależny od wszystkich pozostałych bitów wiadomości, z wyjątkiem interferencji nieliniowego przenoszenia, możemy wyrazić dowolny kluczowy bit pod względem bitów znanej wiadomości przez

$$\begin{aligned} K_1[i] &= L_{64}[i] \oplus R_{64}[i] \oplus L_0[i] \oplus C_1[i], \\ K_2[i] &= L_0[i] \oplus R_0[i] \oplus R_{64}[i] \oplus C_2[i], \end{aligned}$$

gdzie i jest pozycją bitu, a C_1 i C_2 są nieliniowymi funkcjami wiadomości i kluczowymi bitami, ze względu na przenoszenie. Teraz rozważmy następny bit, bit 1, na który wpływa przeniesienie wartości frombitowej 0. Jeśli wyeliminujemy przeniesienie z bitu 1, pozostawiamy część liniową odpowiadającą równaniu 15, które można rozwiązać, aby odzyskać dwa kolejne klucze bitowe. Kontynuujemy w ten sposób, aż cały klucz zostanie odzyskany. Najprostszym sposobem na wyeliminowanie przenoszenia jest obliczenie jego wartości dla odpowiedniego bitu, a następnie na wyłączność - lub na to z bitem szyfrogramu. Aby obliczyć przeniesienie dla i-tego bitu, zaszyfruj (i - 1) najmniej znaczące bity tekstu jawnego, używając znanych (i - 1) bitów klawiszy. Wartością bitu w i-tej pozycji jest wartość przeniesienia (którą wyłączamy - lub w stosunku do faktycznego bitu tekstu zaszyfrowanego). W ten sposób możemy szybko odzyskać klucz 64-bitowy za pomocą tylko jednego znanego tekstu jawnego i 32 zaszyfrowań. Ogólnie, dla długości bloku wiadomości m, atak ten odzyska klucz m-bitowy z tylko m 2 kodowaniami, niezależnie od liczby rund. Ataku tego nie można zastosować do TEA, ponieważ zmiany w funkcji szyfrowania zapewniają lepszą dyfuzję. Gdy każda runda przechodzi, każdy bit bloku wiadomości staje się bardziej zależny od innych wiadomości i kluczowych bitów. Każdy bit zaszyfrowanego tekstu TEA jest złożoną nieliniową funkcją wielu jawnych i kluczowych bitów; nie ma prostego sposobu rozwiązania równań. Brak dyfuzji w STEA oznaczał, że można znaleźć zależność

liniową między wiadomościami i kluczowymi bitami w tej samej pozycji; i to była oczywiście jego pięta Archilles.

8 Liniowa kryptoanaliza

Metoda liniowej kryptoanalizy jest jedną z najnowszych technik analizy szyfrów blokowych. Jest to atak znanego prostego tekstu (statystycznego), opracowany przez Mitsuru Matsui, używany do łamania szyfru DES przy użyciu 50 stacji roboczych i 2^{43} znanych tekstów jawnych. Liniowa kryptoanaliza przybliża nieliniową część szyfru do równania liniowego (tak, że liniowy szyfr daje większość wyników w postaci nieliniowego szyfrowania przez większość czasu, ale daje również pewne nieprawidłowe wyniki). Ponieważ aproksymacja liniowa ma jedynie charakter probabilistyczny, potrzebujemy wielu znanych tekstów jawnych, aby móc zastosować przybliżenie, szczególnie jeśli prawdopodobieństwo przybliżenia jest skrajnie bliskie połowie (w tym przypadku przybliżenie zapewnia jedynie niewielką przewagę nad przypadkowo zgadywanie wartości bitu klucza). Istnieją dwa etapy zastosowania kryptoanalizy liniowej do szyfru blokowego: (1) znalezienie odpowiednich przybliżeń liniowych szyfru i (2) zastosowanie znanego algorytmu ataku tekstowego. Przegląd, w jaki sposób można znaleźć odpowiednie przybliżenia liniowe dla szyfru Feistela, przedstawia się następująco:

Krok 1 Znajdź równania liniowe, które są dobrymi przybliżeniami do nieliniowej części funkcji szyfrowania. Zwróć uwagę na prawdopodobieństwo, z jakim aproksymacja liniowa się utrzymuje.

Krok 2 Rozszerz liniowe przybliżenia do funkcji okrągłej, a tym samym sformułuj równanie liniowe dla każdego przybliżenia.

Krok 3 Zbuduj liniowe przybliżenie szyfru blokowego poprzez połączenie równań liniowych dla funkcji rundy, upewniając się, że wszystkie pośrednie nieznane terminy wiadomości są anulowane. Prawdopodobieństwo tego przybliżenia liniowego szyfru można obliczyć na podstawie prawdopodobieństw okrężnych przybliżeń.

Krok 4 Oblicz liczbę znanych zwykłych tekstów wymaganych do znanego ataku tekstem jawnym. To zależy od prawdopodobieństwa przybliżenia szyfru, a także od wymaganego stopnia powodzenia.

Aby zilustrować, co rozumiemy przez przybliżenie liniowe, zbadamy właściwości trójprzewodowej nieliniowej funkcji boolowskiej F, zdefiniowanej przez

$$F(A,B,C) = ABC \oplus BC \oplus B \oplus C. \qquad (16)$$

Poniższa tabela pokazuje dane wyjściowe F dla wszystkich możliwych danych wejściowych:

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Zauważ, że ta funkcja nieliniowa jest stronnicza: wyjście ma więcej niż zero. Nierozsądne jest wykorzystywanie stronniczych funkcji jako nieliniowego składnika szyfru blokowego (lub rzeczywiście dowolnego systemu szyfrowania). Ta stronniczość z pewnością wyciekłaby z niektórych informacji na temat jawnego tekstu i wprowadziła szyfrogram; i może to prowadzić do ataku wybranego lub znanego-jawnego tekstu. Transformacja Walsha służy do pomiaru liniowości funkcji boolowskiej. W szczególności dla nieliniowej funkcji F: $\mathbb{F}^n_2 \rightarrow \mathbb{F}_2$, obliczamy

$$S_F(\alpha) = \sum_{x \in F_2^n} (-1)^{x \cdot \alpha} F(x), \qquad (17)$$

gdzie x · α oznacza wewnętrzny produkt wektorów x i α . Wektor α jest n-tką bitów wejściowych, które są testowane pod kątem korelacji z bitem wyjściowym. Wynik równania 17 jest liczbą całkowitą z zakresu -2^{n-1} do 2^{n-1}, i będzie równy tylko wtedy, gdy funkcja nieliniowa jest zrównoważona 0-1 (wyjście ma tę samą liczbę zer i jedynek). W szczególności wynik jest mniejszy od zera, jeśli korelacja jest zachowana z prawdopodobieństwem większym niż 1/2, zero, jeśli nie ma korelacji, i większym od zera, jeśli korelacja utrzymuje się z prawdopodobieństwem mniejszym niż 1/2. Podczas analizowania DES-Boxów, Matsui uogólnia transformację Walsha do funkcji N_F: $\mathbb{F}^n_2 \rightarrow \mathbb{F}_2^m$, który mierzy liniowość między podzbiorami wejść a wyjściami funkcji nieliniowej (transformacja Walsha ma miejsce, gdy funkcja nieliniowa ma jedno-bitowe wyjście). Jeśli transformacja Walsha jest obliczana na równaniu 16 na wszystkich możliwych wartościach?, Pokazuje, że istnieje siedem wyraźnych aproksymacji liniowych (równania liniowe, których wynik jest skorelowany z wyjściem funkcji nieliniowej), jak następuje:

$$\text{Prob=}5/8 \quad : \quad C,$$

$$\begin{aligned}
\text{Prob=}5/8 &: B, \\
\text{Prob=}7/8 &: B \oplus C, \\
\text{Prob=}5/8 &: A \oplus 1, \\
\text{Prob=}5/8 &: A \oplus C, \\
\text{Prob=}5/8 &: A \oplus B, \\
\text{Prob=}5/8 &: A \oplus B \oplus C \oplus 1.
\end{aligned}$$

Można je zweryfikować porównując ich wyniki z wynikami równania nieliniowego

A	B	C	F	$B \oplus C$	$A \oplus 1$	$A \oplus C$	$A \oplus B$	$A \oplus B \oplus C \oplus 1$
0	**0**	**0**	**0**	**0**	1	**0**	**0**	1
0	0	**1**	**1**	**1**	**1**	**1**	0	0
0	**1**	0	**1**	**1**	**1**	0	**1**	0
0	**1**	**1**	**1**	0	**1**	**1**	**1**	**1**
1	**0**	**0**	**0**	**0**	**0**	1	1	**0**
1	0	**1**	**1**	**1**	0	0	**1**	**1**
1	**1**	0	**1**	**1**	0	**1**	0	**1**
1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
3	5	5	**8**	7	5	5	5	5

Najwyraźniej najlepszym przybliżeniem nieliniowej ekspresji ABC ⊕ BC ⊕ B ⊕ C jest liniowa ekspresja B ⊕ C, który daje wynikowy wynik dla 7 z 8 możliwych wejść. Wykażemy teraz liniową kryptoanalizę prostego 4-okrągłego szyfru Feistela zgodnie z omówionym wcześniej przeglądem. Niech funkcja szyfrowania zostanie zdefiniowana przez

$$F(M,(Y,Z)) \quad = \quad MY \oplus MYZ \oplus Z,$$

gdzie Y i Z są połówkami okrągłego podklucza. Zatem proces szyfrowania to

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus R_{i-1}Y_i \oplus R_{i-1}Y_iZ_i \oplus Z_i, \end{aligned} \tag{18}$$

dla $1 \le r \le 4$. Tekst jawny $P = (L_0, R_0)$ i tekst zaszyfrowany $C = (R_4, L_4)$. Może to zostać zaatakowane przez podejście "dziel i rządź", ponieważ każdy bit szyfrogramu zależy od jednego bitowego tekstu jawnego, a każdy bit komunikatu jest szyfrowany niezależnie (nie ma dyfuzji). Zastosowanie permutacji bitowej po funkcji szyfrowania naprawiłoby brak dyfuzji, ale w tym przypadku nie chroniłoby przed kryptoanalizą liniową.

Krok 1. Po jednej rundzie szyfrowania nieznany blok wiadomości to (L_1, R_1), gdzie L_1 jest znanym tekstem jawnym, a $R_1 = L_0 \oplus R_0Y_1 \oplus R_0Y_1Z_1 \oplus Z_1$. W szczególności R_1 jest cztero-wejściową nieliniową funkcją postaci $A \oplus B \oplus BCD \oplus CD$. W rzeczywistości ta funkcja nieliniowa jest zrównoważona 0-1. Transformacja Walsha ujawnia osiem liniowych przybliżeń:

$$\begin{aligned} \text{Prob=10/16} &: A \oplus 1, \\ \text{Prob=10/16} &: A \oplus D, \\ \text{Prob=10/16} &: A \oplus C, \\ \text{Prob=10/16} &: A \oplus C \oplus D \oplus 1, \\ \text{Prob=14/16} &: A \oplus B, \\ \text{Prob=10/16} &: A \oplus B \oplus D, \\ \text{Prob=10/16} &: A \oplus B \oplus C, \\ \text{Prob=10/16} &: A \oplus B \oplus C \oplus D \oplus 1. \end{aligned}$$

Zatem najlepsze przybliżenie liniowe jest po prostu

$$A \oplus B \oplus BCD \oplus CD \quad \approx \quad A \oplus B$$

która posiada z prawdopodobieństwem 14 / 16 = 0,875.

Krok 2. Poprzez dopasowanie terminów funkcji rundy (równanie 18) do wyrażenia A ⊕ B ⊕ BCD ⊕ CD i zastępując w proponowanym aproksymacji liniowej A ⊕ B, otrzymujemy okrągłą liniową aproksymację ℓ-tego bitu wyjściowego

$$\begin{aligned} L_i[\ell] &= R_{i-1}[\ell], \\ R_i[\ell] &= L_{i-1}[\ell] \oplus Z_i[\ell], \end{aligned}$$

To jedno-rundowe równanie liniowe odpowiada wynikowi rzeczywistej nieliniowej funkcji rundy z prawdopodobieństwem 0.875 (to jest to samo, co przybliżenie liniowe). Należy zauważyć, że prawdopodobieństwo tego przybliżenia liniowego dotyczy tylko jednego bitu, a nie całego bloku. Z tego powodu położenie bitów w przybliżeniu liniowym jest zwykle również zapisywane w równaniu. Jest to konieczne, jeśli różne funkcje nieliniowe są obliczane na sublokacjach, tak jak w polach DES S-box, lub gdy w szyfrze stosowane są kombinacje bitowe. Pominiemy notację, ponieważ poniższa analiza dotyczy tylko bitów w tej samej pozycji w bloku.

Krok 3. Jeśli mamy więcej niż jedno równanie liniowe dla tej rundy, ten krok wymagałby dodania równoległych równań liniowych, tak aby bity wyjściowe jednego równania stały się wejściowymi bitami wiadomości późniejszej rundy, tak aby ostatecznie zostały anulowane. Powstałe w ten sposób równanie reprezentowałoby szyfr Feistela (oczywiście trzymając probabilistycznie) i wyrażało liniową zależność między niektórymi bitami tekstu jawnego, klucza i tekstu zaszyfrowanego. Ponieważ mamy tylko jedno równanie liniowe, ten krok po prostu wymaga rozszerzenie szyfru Feistela pod względem równania jednokrotnego. Ponownie tekst jawny to P = (L_0 , R_0), a tekst zaszyfrowany to C = (R_4, L_4). Łącząc jednokrotne przybliżenia liniowe uzyskujemy:

$$L_0 \oplus R_0 \oplus L_4 \oplus R_4 \quad = \quad Z_1 \oplus Z_2 \oplus Z_3 \oplus Z_4. \qquad (19)$$

Prawdopodobieństwo, że to równanie zachodzi dla losowego znanego tekstu jawnego, zależy od prawdopodobieństw każdej liniowej aproksymacji, z której się składa. Przybliżenie liniowe za pomocą szyfrów zachowuje się wtedy i tylko wtedy, gdy zachowane zostaną wszystkie jednokierunkowe aproksymacje liniowe lub jeśli żadne z nich nie zostanie zatrzymane. W tym przypadku prawdopodobieństwo równania 19 jest obliczane przez

$$\begin{aligned} & (0.875)(0.875) + (1-0.875)(1-0.875) \quad = \quad 0.781 \\ \rightarrow\ & (0.781)(0.875) + (1-0.781)(1-0.875) \quad = \quad 0.711 \\ \rightarrow\ & (0.711)(0.875) + (1-0.711)(1-0.875) \quad = \quad 0.658 \end{aligned}$$

Metoda uzyskiwania prawdopodobieństwa przybliżenia liniowego szyfru z prawdopodobieństw jednokrokowej liniowej przybliżenia w ten sposób wynikają z pasterstwa lematu opisanego przez Matsui. Równoważnie, prawdopodobieństwo przybliżenia liniowego szyfru można obliczyć o

$$\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \left(p_i - \frac{1}{2} \right)$$

która ocenia prawdopodobieństwo, że wyłączna lub n niezależnych zmiennych losowych będzie równa zero. Nie można użyć ślepego lematu; nie jest w stanie podać dokładnego oszacowania prawdopodobieństwa dla niektórych szyfrów blokowych, szczególnie w przypadku permutacji z kluczem lub permutacji zależnych od danych.

Krok 4. Sukces ataku zależy od prawdopodobieństwa przybliżenia liniowego szyfru, a także od liczby znanych jawnych tekstów. Matsui szacuje wskaźnik sukcesu algorytmu i liczbę zwykłych tekstów wymaganych do osiągnięcia określonego poziomu sukcesu. Wskaźnik skuteczności opiera się na metodzie największej wiarygodności i jest uzyskiwany poprzez przybliżenie rozkładu binarnego z rozkładem normalnym. Dla 97,7% skuteczności sugerowane jest równanie $N = |p-1/2|^{-2}$. Zatem, dla wskaźnika sukcesu 97,7%, potrzebowalibyśmy $|0.658 – 0.500|^{-2} = 40$ znanych jawnych tekstów. Jeśli liczba rund zostanie zwiększona z czterech do ośmiu, prawdopodobieństwo przybliżenia liniowego

szyfru wynosi 0.550, a około 400 znanych tekstów jawnych jest wymaganych dla 97,7% skuteczności. Aby wykonać atak znanym tekstem jawnym na ten 4-okrągły szyfr Feistela, uzyskaj N = 40 znanych tekstów jawnych (zaszyfrowanych pod nieznanym kluczem). Niech X = 0, dla każdego znanego tekstu jawnego, oceń jednobitową wartość lewej strony (LHS) przybliżenia liniowego szyfru (równanie 19) i zwiększ licznik X o jeden, jeśli LHS ma wartość zero. Jeśli X > N/2, to zgadnij, że wartość prawej strony równania 19 wynosi zero, inaczej zgadnij, że jest to jedna. Daje nam to jedno-bitowe rozwiązanie dla $Z_1 \oplus Z_2 \oplus Z_3 \oplus Z_4$, które będzie używane do ustalania wartości bitu klucza w ekscytującym wyszukiwaniu klucza (zmniejszając w ten sposób o połowę złożoność wyczerpującego wyszukiwania klucza). Jest to podstawowy znany atak w postaci zwykłego tekstu; istnieją ulepszenia, które próbują odzyskać większą część klucza naraz i wymagają mniej znanego tekstu jawnego. Składnik nieliniowy w STEA (i TEA) jest operacją dodawania całkowitoliczbowego ($\boxplus$). Nieliniowość w dodawaniu całkowitym wynika z funkcji przenoszenia, jak pokazano w równaniu 14. Funkcja S [i] = (A $\boxplus$ B) [i] (tj. algebraiczna normalna postać S [i], w której S = A? B) staje się bardziej złożony, ponieważ się zwiększa, z uwagi na fakt, że niesie propagację w górę. Poniższa tabela pokazuje działanie funkcji przenoszenia:

$A[i]$	$B[i]$	$C[i]$	$C[i+1]$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Transformacja Walsha ujawnia cztery liniowe przybliżenia funkcji przenoszenia, każde z prawdopodobieństwem 6 = 8 = 0,750:

$$\begin{aligned}
\text{Prob=6/8} &: C[i], \\
\text{Prob=6/8} &: B[i], \\
\text{Prob=6/8} &: A[i], \\
\text{Prob=6/8} &: A[i] \oplus B[i] \oplus C[i] \oplus 1.
\end{aligned}$$

Daje nam to dwie użyteczne przybliżenia liniowe dla jednej operacji przenoszenia, C [i] = A [i - 1] i C [i] = B [i - 1], obie utrzymujące z prawdopodobieństwem 0.750. Odpowiednie jednokierunkowe przybliżenia liniowe są

$$R_i[\ell] = L_{i-1}[\ell - 1], \tag{20}$$

$$R_i[\ell] = R_{i-1}[\ell - 1] \oplus K_i[\ell - 1], \tag{21}$$

Aby znaleźć przybliżenie liniowe dla szyfru z 64 rundami, równania te muszą zostać rozszerzone na odpowiednią liniową ścieżkę od tekstu jawnego do tekstu zaszyfrowanego. Optymalną konstrukcją dla ścieżki liniowej w STEA byłoby użycie równania 20 dla wszystkich rund nieparzystych i równania 21 dla wszystkich równych rund. Wynikiem jest przybliżenie liniowe STEA

$$L_{64}[32] \oplus R_{64}[32] \quad = \quad K_1[31]$$

która jest nieważna, ponieważ bit 32 nie istnieje dla bloków 32-bitowych. Liniowa kryptoanaliza jest nieskuteczna w odniesieniu do szyfru STEA, ponieważ nie ma liniowej aproksymacji, która obejmuje 64 rundy. Ogólnie, najlepsze przybliżenie liniowe w n-rundzie STEA daje

$$L_n[n/2] \oplus R_n[n/2] \quad = \quad K_1[(n/2) - 1].$$

Stąd dla 16-rundowej STEA mamy aproksymację liniową

$$L_{16}[8] \oplus R_{16}[8] \quad = \quad K_1[7]$$

który występuje z prawdopodobieństwem 1/2 + 2^{-17}. Aby odzyskać jeden bit klucza (bit 7 z K_1) 16-rundowej STEA przy użyciu kryptoanalizy liniowej wymagałoby około 2^{34} znanych jawnych tekstów.

9 Słaby Kluczowy Harmonogram i słaba funkcja szyfrowania

W tej sekcji pokazujemy, jak ważne jest posiadanie kluczowego harmonogramu, który generuje różne okrągłe podklucze. Najpierw wykonamy analizę z części 5 i opiszemy ciekawą właściwość szyfru STEA. Jeśli najmniej znaczące bity podkluczy i stron w postaci zwykłego tekstu są równe, to te bity będą "filtrować" do zaszyfrowanego tekstu. Na przykład,

$$\begin{aligned} P &= (\texttt{xxxxxyyy xxyyyyyy}_{\mathrm{H}}), \\ K &= (\texttt{xxxxyyyy xxxxxxyy}_{\mathrm{H}}), \\ C &= (\texttt{xxxxxyyy xxxxxxyy}_{\mathrm{H}}), \end{aligned}$$

gdzie "x" oznacza dowolną cyfrę, a "y" oznacza sekwencję bitów wszystkich (F_H) lub zero (0_H). Powinno to być oczywiste w przypadku, gdy y = 0, od dodania i wyłączności - lub też skutkuje 0 i brakiem przenoszenia. Ponadto, jeśli K_1 jest równy prawej połowie tekstu jawnego, a K_2 jest równy lewej połowie tekstu jawnego, tekst jawny nie jest szyfrowany. Jeśli atakujący ma dużą liczbę bloków tekstu zaszyfrowanego, a jeden blok wygląda podejrzanie jak zwykły tekst, to może być kluczem. Ten problem spowodowany jest przez wyjście funkcji szyfrowania (równanie 10) równe zero dla wszystkich rund. Wyjście funkcji szyfrowania jest równe zero wtedy i tylko wtedy, gdy blok komunikatu wejściowego jest równy podkluczowi. Ponieważ blok wiadomości nie może być zaszyfrowany, jeśli wyjście funkcji szyfrowania jest równe zero, ważne jest, aby następny podklucz, który wpływa na blok komunikatów, był inny, aby zapewnić, że wynik funkcji szyfrowania jest niezerowy (i spowodować, że blok komunikatu będzie częściowo zaszyfrowany, jako wymagane). W raporcie badawczym IBM Edna Grossman i Bryant Tuckerman opisali szereg ataków na słaby szyfr Feistela o nazwie NDS (New Data Seal), który wydaje się być uproszczoną wersją szyfru blokowego Lucifera . Szyfr NDS miał słaby harmonogram i słabą funkcję szyfrowania. Kluczowy harmonogram praktycznie nie istniał; wszystkie podklucze były takie same, co skutecznie oznaczało, że w każdej rundzie cały klucz jest wprowadzany w tej samej formie (tj. niezmodyfikowany między rundami). Funkcja szyfrowania była słaba z dwóch powodów: zazwyczaj możliwe było wyprowadzenie kluczowych bitów ze znajomości bloku wejściowego i (zaszyfrowanego) bloku wyjściowego funkcji szyfrowania; ponadto możliwe byłoby przewidzenie wyjścia funkcji szyfrowania dla niektórych wejść, niezależnie od klucza. W konsekwencji NDS okazał się wysoce podatny na atak, który w swojej najbardziej zaawansowanej formie mógł złamać NDS 556 wybranym jawnym tekstem. Jednak zasada ich ataku jest dość ogólna i mogłaby mieć zastosowanie do dowolnego

szyfru Feistela, który nie używa różnych podkluczy, a którego funkcja szyfrowania może umożliwić odzyskiwanie kluczowych bitów ze znanych bloków wejściowych i wyjściowych. Ciekawym punktem na temat tego ataku jest to, że jego skuteczność nie zależy od liczby rund. Podstawowa zasada ich ataku opiera się na fakcie, że proces szyfrowania n-okrągłego szyfrowania Feistela bez wyraźnych okrągłych podkluczy może zostać uproszczony do $E(P,K) = \pi(\rho^n(P,K))$. W poniższym objaśnieniu pominąć transformację wyjściową (π) Dla uproszczenia, chociaż analiza nadal ma zastosowanie do szyfrów Feistel z ostateczną zamianą. Dość łatwo jest przedłożyć dowody, aby uwzględnić ostateczną zamianę. Uzyskamy znany tekst jawny (P, C), gdzie $C = \rho^n(P,K)$, i przewidujemy wynik jedno-okrągłego szyfrowania P; oznaczamy to przez P', aby $P' = \rho(P, K)$. Ponieważ mamy do czynienia z szyfrem Feistela, wiemy, że $P = (L_0, R_0)$ i $P' = (L_1, R_1)$, gdzie tylko prawa połowa P' jest nieznana i musi zostać przewidziana. Przewidywanie wyniku jednokrotnego zaszyfrowania NDS było łatwe ze względu na słabą funkcję szyfrowania. W rzeczywistości funkcja szyfru NDS jest znacznie słabsza niż funkcja szyfrująca STEA (nie można przewidzieć wyjścia funkcji szyfrującej STEA, jeśli okrągły podklucz nie jest znany). Jeśli nie można przewidzieć wyjścia funkcji szyfrowania, a długość bloku komunikatu jest wystarczająco mała (np. M = 64), wówczas możemy wypróbować każdą możliwą wartość R_1, dopóki nie znajdziemy właściwej wartości; atak wymagałby najwyżej $2^{m/2}$ wybranych szyfrowanych tekstów. Otrzymujemy wybrane szyfrowanie jawnego P' , aby uzyskać $C' = \rho^n(P', K)$. Zauważ, że C' oznacza (n + 1) - całkowite zaszyfrowanie P, tj. $C' = \rho(C, K)$:

$$\begin{aligned} C' &= \rho^n(P', K) \\ &= \rho^n(\rho(P, K), K) \\ &= \rho^{n+1}(P, K) \\ &= \rho(\rho^n(P, K), K) \\ &= \rho(C, K). \end{aligned}$$

W szczególności, dla poprawnego odgadnięcia $P' = \rho(P,K)$, prawa połowa C powinna być równa lewej połowie C', ponieważ $C = (L_n , R_n)$ i $C' = (L_{n+1}; R_{n+1})$. Po znalezieniu właściwej wartości $P' = \rho(P, K)$, możemy spróbować odzyskać klucz K z równań:

$$\begin{aligned} L_0 \oplus R_1 &= F(R_0, K), \\ L_n \oplus R_{n+1} &= F(R_n, K). \end{aligned}$$

Teraz problem polega na odzyskaniu klucza, biorąc pod uwagę wejście i wyjście funkcji szyfrowania. Ponieważ mamy dwa takie równania, możemy użyć jednego równania, aby określić klucz, a następnie zweryfikować klucz, który odzyskaliśmy, testując go w drugim równaniu. Jeśli funkcja szyfrowania jest słaba, odzyskanie klucza, biorąc pod uwagę funkcję wejścia i wyjścia funkcji szyfrowania, powinno być stosunkowo proste (jak w przypadku NDS lub STEA). Wariant tego ataku można zastosować do STEA przy wyższym koszcie obliczeniowym. STEA tego nie robi ,używaj tego samego podklucza w każdej rundzie, ale używa tego samego podklucza w kolejnych rundach. Niech K_1 i K_2 będą 32-bitowymi połówkami klucza 64-bitowego K. Możemy wyrazić powtarzalny charakter STEA, pozwalając R oznaczać dwie rundy:

$$R(M, K) = \rho(\rho(M, K_1), K_2)$$

Stąd 64-rundy STEA może być reprezentowane jako 32 kolejne aplikacje funkcji R. W szczególności klucz nie zmienia się między zastosowaniami R tak, że E(P,K) = R^{32}(P,K). Musimy teraz przewidzieć wynik R dla wybranego tekstu jawnego. Funkcja szyfrowania STEA nie pozwala nam tego zrobić; musimy znać wartość podklucza, aby przewidzieć wynik jednej rundy. Jedynym sposobem radzenia sobie z tą sytuacją jest uzyskanie wystarczająco rozpowszechnionego znanego tekstu jawnego, że mamy uzasadnioną szansę znalezienia pary tekstów jawnych (P, P') w taki sposób, że P' = R(P,K). W wyniku urodzinowego paradoksu wymagałoby to około 2^{32} znanych tekstów zwykłych. Ponieważ nie możemy zidentyfikować prawidłowej pary (P, P'), musimy wykonać następującą analizę każdej możliwej pary, aż do odzyskania klucza. Dlatego najgorszym przypadku złożoność tego ataku byłaby O(2^{64}). Jest to porównywalne ze złożonością wyczerpującego wyszukiwania kluczy. Dla każdej pary znanych jawnych tekstów (P, P') rozwiązujemy następujące równania

$$
\begin{aligned}
K_1 &= R_0 \oplus (L_2 \boxminus L_0), \\
K_2 &= L_2 \oplus (R_2 \boxminus R_0), \\
K_1 &= R_{64} \oplus (L_{66} \boxminus L_{64}), \\
K_2 &= L_{66} \oplus (R_{66} \boxminus R_{64}),
\end{aligned}
$$

gdzie P = (L_0, R_0), P' = (L_2, R_2), C = (L_{64},R_{64}) i C' = (L_{66}, R_{66}). Jeżeli dwie wartości K_1 są zgodne i dwie wartości K_2 są zgodne, to znaleźliśmy właściwy klucz. Po raz kolejny atak ten będzie działał niezależnie od liczby rund i może nadal być skuteczny, jeśli funkcja szyfrowania jest nieco bardziej złożona (np. Jeśli wyjście funkcji szyfrowania zostało poddane obrotowi bitowemu, co spowodowałoby podział - i - konkurencyjny atak nieskuteczny). Podobny atak można wykonać na wersji TEA z ustalonymi stałymi podklucza δ_i(np. δ_i = δ, Ddla wszystkich i); ta wersja TEA miałaby dwa odrębne podklucze (jak STEA). Atak na ten wariant TEA został opisany przez Rogera Fleminga . Ma złożoność O (2^{98}) i wymaga około 2^{32} znanych tekstów jawnych. Tutaj złożoność ataku jest lepsza niż w przypadku wyczerpującego wyszukiwania kluczy (co nie miało miejsca w przypadku STEA). Udoskonalenie ataku, który redukuje złożoność do O (2^{64}) kosztem wymagania 232 wybranych jawnych tekstów, zostało nakreślone przez Davida Wagnera.

10 Kryptoanaliza różnicowa

W części 2 pokazano, że trudność rozwiązywania równań nieliniowych może być przydatna do projektowania szyfrów blokowych. Na przykład, wróćmy do bloku szyfrowego równania 3. Najdalej mogliśmy uzyskać poprzez proste manipulowanie tym szyfrem, eliminując podklucz, przyjmując różnicę zaszyfrowanych tekstów, dochodząc do równania 4. Zauważmy teraz, że jeśli szyfr był liniowy , przyjęcie "różnicy" pary zaszyfrowanych tekstów spowodowałoby anulowanie obu podkluczy, pozostawiając równanie:

$$C \oplus C' = P \oplus P'$$

To po prostu mówi nam, że różnica między szyfrogramami jest taka sama jak różnica między ich tekstami jawnymi; i jak to ma miejsce w przypadku wszystkich wiadomości, uwzględnienie różnicy nie prowadzi do żadnej informacji o kluczu. W przypadku nieliniowego szyfru różnica między zaszyfrowanymi tekstami nie jest taka sama jak różnica między ich zwykłym tekstem dla wszystkich możliwych tekstów jawnych. Ponadto, dla określonej różnicy w zwykłym tekście, różnica w odpowiadającym zaszyfrowanym tekście może zależeć od klucza, a zatem ujawnić informacje o kluczu. Idea ta jest zasadą kryptoanalizy różnicowej, ogólnym analitycznym algorytmem blokowym opracowanym przez Eli Bihama i Adi Shamira. Pomyślne zastosowanie kryptoanalizy różnicowej

prowadzi do ataku na wybrany tekst jawny. Oznaczamy różnicę między parą wiadomości (M , M') przez $\Delta M = M \otimes (M')^{-1}$, gdzie jest odpowiednia operacja do definiowania "różnicy". Ekskluzywność - lub działanie, ze swej natury, jest najbardziej uniwersalnym wyborem do zdefiniowania różnicy. Jednak w tym celu może być korzystne lub konieczne użycie innej operacji grupowej; na przykład, gdy stosuje się kryptoanalizę różnicową do iterowanego szyfru, zaleca się użycie operacji różnicowej, która spowoduje, że szyfr stanie się "szyfrem Markowa". Przykładowy szyfr blokowy równania 3 nie jest iterowanym szyfrem blokowym, ale jest odpowiedni do demonstrowania podstawowych idei kryptoanalizy różnicowej. Przyjmujemy długość bloku wiadomości m = 32 i długość klucza k = 64 (czyli połowę długości bloku oryginalnego opisu). Różnica pozwalająca zdefiniować przez odjęcie modulo 2^{32}, następujące równanie daje odpowiednią różnicę w szyfrogramie:

$$\begin{aligned} \Delta C &= C \boxminus C' \\ &= (P \oplus K_0) \boxminus (P' \oplus K_0). \end{aligned}$$

Stąd atakujący, który zna wartości P, P' i ΔC może odzyskać K_0, jeśli potrafi efektywnie rozwiązywać równania w postaci

$$C = (A \oplus X) \boxminus (B \oplus X)$$

gdzie A, B i C są znanymi stałymi. Zbadajmy więc ten typ równania bardziej szczegółowo, aby dokładnie określić, ile informacji możemy odzyskać. Najpierw wyrażamy równanie w prostszej formie:

$$\begin{aligned} C &= (A \oplus X) \boxminus (B \oplus X) \\ &= (A \oplus X) \boxminus (A \oplus X \oplus D) \\ &= Y \boxminus (Y \oplus D), \end{aligned}$$

gdzie $D = A \oplus B$ i $Y = A \oplus X$. Celem jest odzyskanie Y. Odejmowanie modulo 2^{32} można zastąpić przez dodanie modulo 2^{32} z użyciem małej manipulacji

$$\begin{aligned} C &= Y \boxminus (Y \oplus D) \\ &= Y \boxplus -(Y \oplus D) \\ &= Y \boxplus (Y \oplus D \oplus \mathrm{FFFFFFFF_H}) \boxplus 1 \\ E &= Y \boxplus (Y \oplus F), \end{aligned}$$

gdzie $F = D \oplus \mathrm{FFFFFFFF_H}$ (bitowe uzupełnienie D) i $E = C \boxminus 1$. Ponownie Y jest nieznane (a jego wartość prowadzi do podklucza), a E i F są znane. W ataku wybranym tekstem jawnym napastnik będzie w stanie ustalić wartości F przez staranny dobór jawnych tekstów A i B. Jeśli F = 0, to będziemy mieli $E = Y \boxplus Y = Y \ll 1$, efektywnie odsłaniając 31 z 32 bitów Y. Aby ustawić F = 0, różnica exclusive-or między tekstami jawnymi musi być ($\mathrm{FFFFFFFF_H}$); nie ma znaczenia. Rozważmy atak na wybrany tekst jawny na ten szyfr blokowy. Atakujący wybiera losowy tekst jawny P, a następnie oblicza drugi zwykły tekst $P' = P \oplus \Delta P$ (tak, że wyłączność - lub różnica między tekstami jawnymi to ΔP). Oba wybrane jawne teksty są szyfrowane pod nieznanym kluczem. Najmniej znaczące 31 bitów podklucza K_0 jest następnie odzyskiwane z wartości P i $\Delta C = C \boxminus C'$ przez

$$K_0 = ((\Delta C \boxminus 1) \gg 1) \oplus P.$$

Najbardziej znaczącym bitem odzyskanego K_0 będzie "0". Podklucz K_1 jest następnie oceniany, za pomocą jednego z dwóch wybranych jawnych tekstów, za pomocą $K_1 = C \boxminus (P \oplus K_0)$. W związku z tym klucz 64-bitowy (K_0, K_1) można łatwo odzyskać za pomocą jednego znanego tekstu jawnego i jednego adaptacyjnego wybranego tekstu jawnego (lub dwóch wybranych jawnych tekstów). W tym przykładzie operacja definiowania różnicy była odmienna w przypadku zwykłego tekstu (wyłącznego lub) i tekst zaszyfrowany (odejmowanie modulo 2^{32}). Podejście do analizy iterowanych szyfrów wymaga, aby operacja różnicy była taka sama dla wejścia i wyjścia. W pozostałej części tego raportu zdefiniujemy operację różnicową za pomocą wyłączności - lub. Teraz spróbujemy kryptoanalizy różnicowej na szyfrze blokowej zdefiniowanej przez

$$C = (P \boxplus K_0) \oplus K_1.$$

Szyfr ten różni się od szyfrów blokowych równania 3 tym, że dodanie i operacja exclusive-or zamieniły się miejscami. Operacja różnicy jest definiowana przez exclusive-or, a więc mamy następujące równania dla tekstu jawnego i różnic w szyfrowaniu:

$$\begin{aligned} \Delta P &= P \oplus P', \\ \Delta C &= C \oplus C' \\ &= (P \boxplus K_0) \oplus (P' \boxplus K_0) \end{aligned}$$

Podklucz K_1 został wyeliminowany z różnicy tekstu zaszyfrowanego ΔC. Musimy wybrać wartości P i P', z określoną różnicą ΔP, która ujawni pewne informacje o K0 przez różnicę ΔC. Na początek zwróć uwagę, że jeśli ΔP = 0 to ΔC = 0; nie mamy użytecznych informacji. Należy również zauważyć, że można to uogólnić na przypadek, że sekwencja najmniej znaczących bitów "0" w różnicy wejściowej spowoduje, że te same najmniej znaczące bity będą miały "0" w różnicy wyjściowej, np., Jeśli? P = $xxxx0000_H$ następnie C = $xxxx0000_H$. Dzieje się tak dlatego, że nieliniowe przenoszenie propaguje tylko w górę i, znowu, wykorzysta ten fakt. Waga Hamminga liczby całkowitej jest liczbą jedynek w jej reprezentacji binarnej. Przyjrzymy się teraz, co się dzieje, gdy najmniej znaczący bit wejściowej różnicy jest niezerowy. Należy zauważyć, że ponieważ niższe bity różnicy wejściowej zero nie wpływają na wyższe bitwy różnicy wyjściowej, analiza zachowuje dla każdej różnicy wejściowej, w której dokładnie ustawiony jest jeden bit (to jest, każda różnica wejściowa wagi Hamminga 1). Gdyby różnica wejściowa ΔP = 1 następnie byłaby

$$\Delta C = (P \boxplus K_0) \oplus ((P \oplus 1) \boxplus K_0)$$

Z tego wynika, że najmniej znaczący bit K0 jest dodawany do każdego z "0" i "1", a następnie wyłączna - lub różnica jest podejmowana; wartość najmniej znaczącego bitu P nie ma wpływu na te obliczenia. LSB z ΔC ma wartość "1", ponieważ jest liniowa. Jeśli LSB z $K_0 = 0$, wtedy żadne przeniesienie nie będzie generowane przez żaden z dodatków, a zatem żaden z wyższych bitów różnicy wyjściowej nie zostanie zmieniony (będą miały wartość zero). W tym przypadku ΔP =ΔC. Jeśli LSB z K1 = 1 to obliczenie $(1 \boxplus 0) \oplus (1 \boxplus 1)$ wygeneruje przeniesienie, które wpłynie na wyższe bity. Spodziewamy się, że w tym przypadku $\Delta P \neq \Delta C$. Możemy więc określić wartość najmniej znaczącego bitu K_0, stosując różnicę wejściową 1 i sprawdzając, czy różnica wyjściowa jest równa różnicy wejściowej. Jeśli różnica wyjściowa . Jeśli różnica wyjściowa jest taka sama jak różnica wejściowa, to przyjmijmy, że LSB dla K0 wynosi "0", w przeciwnym wypadku zakłada się, że jest to "1". Możemy odzyskać dowolny bit K_0 według tej samej

techniki; użyć różnicy wejściowej ciężaru Hamminga 1 (ustawić bit w i-tej pozycji) i sprawdzić masę Hamminga różnicy wyjściowej; jeśli ma wagę Hamminga 1, to przyjmijmy, że i-ty bit K0 wynosi "0", w przeciwnym wypadku zakładamy, że jest to "1". Postępując w ten sposób, możemy odzyskać (m - 1) najmniej znaczące bity K_0. Nie możemy odzyskać wartości najbardziej znaczącego fragmentu K_0 w tej analizie, ponieważ i tak tracimy transport. Ta strata jest jednak rekompensowana przez fakt, że automatycznie odzyskujemy ten bit podczas obliczania wartości K_1 ze znanego zwykłego tekstu (odzyskamy oryginalny klucz lub równoważny klucz). Stąd, dla m-bitowego rozmiaru bloku komunikatów (i klucza 2m-bitowego), ten prosty szyfrujący szyfr szyfrujący jest przerywany przez kryptoanalizę różnicową z jednym znanym tekstem jawnym i m wybranym jawnym tekstem. Możliwe jest (ale znacznie bardziej skomplikowane) rozszerzenie tego ataku na blokowy szyfr równania 3 definiujący różnicę wejściową i różnicę wyjściową przez exclusive-or. Mielibyśmy różnicę wyjściową

$$\Delta C \quad = \quad ((P \oplus K_0) \boxplus K_1) \oplus ((P' \oplus K_0) \boxplus K_1).$$

Wyobraźmy sobie, że dostarczamy różnicę wejściową wagi Hamminga 1, z zestawem i-tym bitem. Powikłanie powstaje, ponieważ nie możemy zagwarantować, że bity wejściowe do dodania, które są niższe niż i-ty bit, mają wartość zero, ponieważ niektóre z nich mogły zostać odwrócone przez exclusive-or z K_0. W szczególności, jeśli (i-1) -ty bit K_0 jest "1", a (i-1) -ty bit K_1 również jest "1", to następne przeniesienie, które zostanie wygenerowane przez ich dodanie, zaburzy wyższe bity , a zatem analiza. Ten problem można rozwiązać za pomocą drobiazgowej analizy, ale nie ma to większego sensu, ponieważ nie można uogólnić wyniku (co byłoby głównym powodem zastosowania wyłączności lub zdefiniowania różnicy wejściowej i wyjściowej w tym przypadku). Istnieją dwa etapy zastosowania kryptoanalizy różnicowej do iterowanego szyfrowania (patrz: kryptoanaliza liniowa):

(1) znajdowanie odpowiednich różnic w szyfrowaniu oraz (2) stosowanie algorytmu ataku wybranego przez zwykły tekst. Teraz zastosujemy kryptoanalizę różnicową do sześcio-rundowego szyfrowania blokowego o długości bloku komunikatu m = 8 i długości klucza k = 48. Niech funkcja nieliniowa i odwracalna F: $\mathbb{F}^8_2 \rightarrow \mathbb{F}^8_2$ być określone przez

$$F(x) \quad = \quad ((x \oplus 5C_H) \boxplus 89_H) \ggg 6.$$

Wiemy, że dla funkcji liniowej różnica między dowolną parą wejść będzie równa różnicy między ich odpowiednimi wyjściami. Ponieważ funkcja F jest nieliniowa, różnica wyjściowa nie będzie równa różnicy wejściowej dla wszystkich możliwych wejść. W rzeczywistości, przy specyficznych różnicach wejściowych istnieje wiele możliwych różnic wyjściowych, które mogą wystąpić. Na przykład, niech różnica między wejściami X i X' będzie oznaczona przez ΔX = X⊕X', a różnica ich odpowiednich wyjść Y i Y'0 będzie równa ΔY = F(X) ⊕ F(X'). Odkryliśmy, że istnieje 7 możliwych różnic wyjściowych, które mogą wystąpić, jeśli dane wejściowe różnica do F to 01H, jak następuje (różnice są podane w dwójkowym dla łatwości interpretacji):

* Istnieje 64 par (X, X') takich, że ΔX ? = 00000001 i ΔY = 00001100.

* Istnieje 32 par (X, X') takich, że ΔX = 00000001 i ΔY = 00011100.

* Istnieje 16 par (X, X') takich, że ΔX = 00000001 i ΔY = 00111100.

* Istnieje 8 par (X, X') takich, że ΔX = 00000001 i ΔY = 01111100.

* Istnieją 4 pary (X, X') takie, że ΔX = 00000001 i ΔY = 11111100.

* Istnieją dwie pary (X, X') takie, że ΔX = 00000001 i ΔY = 11111101.

* Istnieją dwie pary (X, X') takie, że ΔX = 00000001 i ΔY = 11111111.

Para różnic wejściowych i wyjściowych dla funkcji tworzy charakterystykę funkcji. Tak więc funkcja F ma charakterystykę (01_H , $1C_H$), która zawiera z prawdopodobieństwem 32/128 = 0.250. Istnieją dwie cechy F, które posiadają prawdopodobieństwo 1. 000:

* Istnieje 128 par (X; X') takich, że ΔX = 00000000 i ΔY = 00000000.

* Istnieje 128 par (X; X') takich, że ΔX = 10000000 i ΔY = 00000010.

Spośród nich tylko ta druga cecha jest użyteczna i wynika z równoważności X $\boxplus 2^{n-1}$ ponad $\mathbb{Z}_{2n}$ i X $\boxplus 2^{n-1}$ ponad $\mathbb{F}^n_2$. Szyfr blokowy iteruje funkcję rund zdefiniowaną przez

$T_i = F(T_{i-1} A \leftrightharpoons \oplus K_i)$, (dla $1 \leq i \leq 6$),

gdzie T_0 jest tekstem jawnym, a T_6 jest tekstem zaszyfrowanym (nie jest to szyfr Feistela). Podklucze (K_1,…,K_6) są niezależne, a ponieważ każdy podklucz ma długość 8 bitów, długość klucza k wynosi 48 bitów. Podczas stosowania kryptoanalizy różnicowej na iterowane szyfry, atakujący analizuje propagację różnic między zwykłymi tekstami za pomocą szyfru blokowego i próbuje znaleźć serię różnic między rundami, które posiadają z dużym prawdopodobieństwem. Odbywa się to poprzez połączenie właściwości jednej rundy, tak aby różnica wyjściowa do jednej rundy stanowiła różnicę wejściową do następnej rundy. Łańcuch różnic dla i rund szyfru tworzy charakterystykę i-rundy.

W przypadku n-rundowego szyfrowania blokowego, atakujący ma nadzieję znaleźć n-okrągłą charakterystykę, która ma duże prawdopodobieństwo. Cecha, która ma optymalne prawdopodobieństwo jest nazywana najlepszą cechą. Zauważ, że dla wejść (X , X') i odpowiednich wyjść (Y; Y 0) występuje jednokierunkowa różnica wejścia ΔX = X ⊕ X' i różnica wyjściowa wynosi ΔY = Y ⊕ Y'. Ponadto

$$\begin{aligned} \Delta Y &= F(X \oplus K) \oplus F(X' \oplus K) \\ &= F(X \oplus K) \oplus F(X \oplus K \oplus \Delta X) \\ &= F(Z) \oplus F(Z \oplus \Delta X), \end{aligned}$$

gdzie Z = X ⊕K jest nieznane. To pokazuje, że nie jesteśmy zainteresowani rzeczywistą wartością danych wejściowych; tylko w ich różnicy. Również umieszcza wcześniejszą analizę funkcji F w kontekście szyfru, ponieważ podklucz jest "pochłonięty" na wejściu funkcji F i nie musimy zajmować się jego wartością dla analizy. Domyślnie zakładamy, że wartość tekstu jawnego i podkluczy nie wpłynie znacząco na propagację różnic za pomocą szyfru (jest to hipoteza stochastycznej równoważności. Wiemy, że najlepszą cechą jednej rundy jest (80_H; 02_H), która posiada prawdopodobieństwo 1.000. Wykorzystamy to jako punkt wyjścia do zbadania możliwych różnic, które mogą wystąpić po 6 rundach. Postaramy się podążać za cechą, która ma najwyższe bezpośrednie prawdopodobieństwo. To niekoniecznie musi prowadzić do najlepszej 6-rundy charakterystyki; musielibyśmy podążać każdą możliwą ścieżką różnicy (włączając różnice o niskim prawdopodobieństwie) lub użyć jakiejś heurystycznej techniki, aby znaleźć najlepszą cechę.

(1) - Istnieje 128 par (X, X') takich, że ΔX = 10000000 i ΔY = 00000010.

(2) - Istnieje 64 par (X, X') takich, że ΔX = 00000010 i ΔY = 00001000.

- Istnieją 32 pary (X, X') takie, że ΔX = 00000010 i ΔY = 00011000.

- Istnieje 16 par (X, X') takich, że ΔX = 00000010 i ΔY = 00111000.

- Istnieje 8 par (X; X0) takich, że ΔX = 00000010 i ΔY = 01111000.

- Istnieją 4 pary (X; X0) takie, że ΔX = 00000010 i ΔY = 11111000.

- Istnieją 2 pary (X, X') takie, że ΔX = 00000010 i ΔY = 11111001.

- Istnieją 2 pary (X, X') takie, że ΔX = 00000010 i ΔY = 11111011.

(3) - Istnieje 16 par (X, X') takich, że ΔX = 00001000 i ΔY = 00100000.

- Istnieje 56 par (X, X') takich, że ΔX = 00001000 i ΔY = 01100000.

- Istnieje 28 par (X, X') takich, że ΔX = 00001000 i ΔY = 11100000.

- Istnieje 14 par (X, X') takich, że ΔX = 00001000 i ΔY = 11100001.

- Istnieje 14 par (X, X') takich, że ΔX = 00001000 i ΔY = 11100011.

(4) - Istnieje 18 par (X, X') takich, że ΔX = 01100000 i ΔY = 10000000.

- Istnieje 92 par (X, X') takich, że ΔX = 01100000 i ΔY = 10000001.

- Istnieje 18 par (X, X') takich, że ΔX = 01100000 i ΔY = 10000010.

(5) - Istnieje 64 par (X, X') takich, że ΔX = 10000001 i ΔY = 00001110.

- Istnieją 32 pary (X, X') takie, że ΔX = 10000001 i ΔY = 00011110.

- Istnieje 16 par (X, X') takich, że ΔX = 10000001 i? Y = 00111110.

- Istnieje 8 par (X, X') takich, że ΔX = 10000001 i ΔY = 01111110.

- Istnieją 2 pary (X, X') takie, że ΔX = 10000001 i ΔY = 11111101.

- Istnieją 4 pary (X, X') takie, że ΔX = 10000001 i ΔY = 11111110.

- Istnieją 2 pary (X, X') takie, że ΔX = 10000001 i ΔY = 11111111.

(6) - Istnieje 16 par (X, X') takich, że ΔX = 00001110 i ΔY = 00001000.

- Istnieje 8 par (X, X') takich, że ΔX = 00001110 i ΔY = 01001000.

- Istnieje 16 par (X, X') takich, że ΔX = 00001110 i ΔY = 01101000.

- Istnieją 32 pary (X, X') takie, że ΔX = 00001110 i ΔY = 01111000.

- Istnieją 4 pary (X, X') takie, że ΔX = 00001110 i ΔY = 11001000.

- Istnieją 2 pary (X, X') takie, że ΔX = 00001110 i ΔY = 11001001.

- Istnieją 2 pary (X, X') takie, że ΔX = 00001110 i ΔY = 11001011.

- Istnieje 8 par (X, X') takich, że ΔX = 00001110 i ΔY = 11101000.

- Istnieją 4 pary (X, X') takie, że ΔX = 00001110 i ΔY = 11101001.

- Istnieją 4 pary (X, X') takie, że ΔX = 00001110 i ΔY = 11101011.

- Istnieje 16 par (X, X') takich, że ΔX = 00001110 i ΔY = 11111000.

- Istnieje 8 par (X, X') takich, że ΔX = 00001110 i ΔY = 11111001.

- Istnieje 8 par (X, X') takich, że ΔX = 00001110 i ΔY = 11111011.

Tak więc nasza 6-okrągła charakterystyka jest

(80H, 02H, 08H, 60H, 81H, 0EH, 78H);

trzymanie z prawdopodobieństwem

128/128 x 64/128 x 56/128 x 92/128 x 64/128 x 32/128 ≈ 1 51

Oznaczałoby to, że na każde 51 losowo i równomiernie rozmieszczonych par wybranych tekstów jawnych z różnica ΔP = 80_H spodziewamy się, że około 1 pary odpowiednich zaszyfrowanych tekstów ma różnicę 78H. Nie jest to jednak tak proste, ponieważ podklucze są stałe (nie losowe) i mogą sprawić, że ścieżki różnicowe znalezione przy pomocy powyższej metody są niemożliwe do zastosowania w praktyce, bez względu na to, jaki mamy wybrany tekst jawny. Ponadto może istnieć więcej niż jedna n-okrągła charakterystyka z tą samą różnicą tekstu prostego i różnicą zaszyfrowanych tekstów. Na przykład, inna 6-okrągła charakterystyka powyższego szyfrowania to

(80_H , 02_H, 18_H , $A3_H$, 86_H , $0A_H$, 78_H);)

który uzyskujemy z prawdopodobieństwem ≈ 1/1561. Nie ma sposobu, aby powiedzieć, która z tych dwóch 6-rundowych cech została wykorzystana do uzyskania różnicy szyfrogramów 78_H od różnicy zwykłego tekstu równej 80H. Ponieważ może być wiele sposobów osiągania określonej różnicy wyjściowej, prawdopodobieństwo uzyskania pary tekstów jawnych, które powodują różnicę wyjściową charakterystyki, może być większe niż prawdopodobieństwo charakterystyki. Atakujący zazwyczaj nie jest zaniepokojony różnicami w komunikatach po każdej rundzie, tylko w ostatniej różnicy charakterystyki. Różnica i-rund to para (ΔX , ΔY) określająca różnicę wejściową i różnicę wyjściową po I rundach. Prawdopodobieństwo określonej różnicy jest sumą prawdopodobieństw wszystkich charakterystyk, które mają taką samą wejściową i końcową różnicę wyjściową, co różnica. Prawa para, w odniesieniu do różnicy odbytu (ΔX , ΔY), jest parą tekstów jawnych, tak że ich różnica jest równa ΔX, a ich różnica po rundach szyfrowania jest równa ΔY. Zła para to para tekstów jawnych, która nie jest odpowiednią parą. Aby wykonać atak kryptoanalizy różnicowej wybranym em jawny na n-rundowy szyfr blokowy:

1. Znajdź (n-1) –rundowy różniczkowy (ΔX , ΔY), który trzyma z dużym prawdopodobieństwem (najlepiej najlepszy (n - 1) -rundowy różniczkowy).

2. Wybierz losowy tekst jawny P i oblicz P' = P ⊕ ΔX. Szyfruj oba jawne teksty tak, aby C = E(P, K) i C'= E(P', K), gdzie K jest nieznanym kluczem.

3. Odzyskujemy wszystkie możliwe wartości podklucza K_n, które mogą spowodować, że jednokierunkowe wyjścia C i C' z wejść z różnicą ΔY. Podstawowym sposobem identyfikowania możliwych podkluczy jest wydedukowanie podkluczy K, które mogą przekształcić wejście Y na wyjście C, dla każdego możliwego wejścia Y, a następnie porównać jedno-rundowe szyfrowanie (Y ⊕ ΔY) z każdym wydedukowanym kluczem w stosunku do zaszyfrowanego tekstu C'. Każdy podklucz K, który przejdzie ten test, jest możliwym kandydatem do faktycznego ostatniego podklucza, a my zwiększamy licznik odpowiadający K o 1.

4. Powtórz test na innym wybranym prostym tekście, aż jedna wartość podklucza zostanie policzona znacznie bardziej niż pozostałe. Oczekuje się, że ten podklucz (lub mały zestaw podkluczy) jest rzeczywistym kluczem użytym w n-tej rundzie.

Jeśli znamy n-rundową różnicę, która obejmuje różnicę (n-1), to znaczy, że będziemy w stanie odrzucić kilka par tekstów jawnych, ponieważ różnice w ich zaszyfrowanych tekstach nie wynikają z różnicy (n - 1). Możemy zastosować kryptoanalizę różnicową do 8-rundowego wariantu STEA. Wariant stosuje lewy obrót w lewo do bloku wiadomości po dodaniu w funkcji rundy. Blok wiadomości jest obrócony o 1 bit w rundach nieparzystych, a 9 bitów w parzystych rundach, tak że funkcja okrągła jest opisana przez

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= (L_{i-1} \boxplus (R_{i-1} \oplus K_i)) \lll S_i, \end{aligned}$$

gdzie $K_i = K_1$ i $S_i = 1$ dla i nieparzystych, a $K_i = K_2$ i $S_i = 9$ dla i. 64-bitowy tekst jawny to (L_0, R_0), a 64-bitowy tekst zaszyfrowany to (L_8, R_8). Obrót w prawo zapewnia lepszą dyfuzję w tym wariancie STEA i pokonałby atak typu dziel i rządź (ale nie liniową kryptoanalizę). Ten STEA ma 6-rundową charakterystykę opisaną przez

$$\begin{aligned} & \texttt{E0000000 C0000000}_H, \\ \text{Prob=0.250} \quad : \quad & \texttt{C0000000 40000000}_H, \\ \text{Prob=0.500} \quad : \quad & \texttt{40000000 00000000}_H, \\ \text{Prob=0.500} \quad : \quad & \texttt{00000000 80000000}_H, \end{aligned}$$

$$\begin{aligned} \text{Prob=1.000} \quad : \quad & \texttt{80000000 00000100}_H, \\ \text{Prob=0.500} \quad : \quad & \texttt{00000100 00000201}_H, \\ \text{Prob=0.125} \quad : \quad & \texttt{00000201 00060200}_H, \end{aligned}$$

który posiada prawdopodobieństwo 1/256. Spodziewamy się, że jedna para 256-parowego tekstu jawnego będzie miała różnicę 00000201 00060200H po 6 rundach szyfrowania. Ponieważ mamy do czynienia z ośmiorundowym szyfrem, nie możemy zidentyfikować właściwych par z tą cechą. Moglibyśmy spróbować ataku na wybrany tekst jawny, gdybyśmy znali szyfrogramy po 7 rundach szyfrowania i możemy je odzyskać przez odszyfrowanie zaszyfrowanego tekstu tylko jedną rundą z każdym możliwym podkluczem K8. Tak więc, dla każdej pary par zaszyfrowanych $C = (L_8, R_8)$ i $C' = (L'_8, R'_8)$, mamy 2^{32} możliwe wartości $(L_7; R_7)$ i $(L'_7, R'7)$. Możemy sprawdzić, czy jest to prawidłowa para, biorąc różnicę lewych połówek tych zaszyfrowanych tekstów i porównując je z oczekiwaną różnicą:

$$L_7 \oplus L'_7 = \texttt{00060200}_H.$$

Jeśli ten warunek nie zostanie spełniony, odrzućmy tę parę i spróbujmy ponownie na innej paryze losowo wybranych tekstów jawnych. W przeciwnym razie zwiększamy licznik dla K_8 i kontynuujemy atak, zakładając, że para jest prawą parą i wyprowadzając możliwe wartości K_7 (jeśli jakieś), dla których

$$L_6 \oplus L'_6 = \texttt{00000201}_H.$$

Wymagałoby to dodatkowych 2^{32} próbnych odszyfrowań dla każdej pary zaszyfrowanych tekstów, które przeszły test od 7. Tak więc złożoność testowania jednej pary wybranych jawnych tekstów wynosi

około 2^{33} i spodziewamy się odzyskać wartości ostatnie dwie rundy podkluczy (i stąd klucz). Metoda kryptoanalizy różnicowej została odkryta przed kryptoanalizą liniową, więc przeprowadzono znacznie więcej badań nad bezpieczeństwem szyfrów blokowych w odniesieniu do kryptoanalizy różnicowej. Istnieje wiele ulepszeń i ewolucji podstawowej idei, takich jak pojęcia skróconych i wyższego rzędu różnic (które zostały użyte do ataku na szyny blokowe IDFER SAFER i zredukowane IDEA)